


For Reference

NOT TO BE TAKEN FROM THIS ROOM



Digitized by the Internet Archive
in 2023 with funding from
University of Alberta Library

<https://archive.org/details/Jatzeck1982>

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR BERNHARD MICHAEL JATZECK, B. SC., MEC. E. (ALTA.)
TITLE OF THESIS NUMERICAL SOLUTION OF SOME SECOND-ORDER DIFFERENTIAL
 EQUATIONS
DEGREE FOR WHICH THESIS WAS PRESENTED MASTER OF SCIENCE
YEAR THIS DEGREE GRANTED FALL 1982

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

THE UNIVERSITY OF ALBERTA

NUMERICAL SOLUTION OF SOME
SECOND-ORDER DIFFERENTIAL EQUATIONS

by



BERNHARD MICHAEL JATZECK, B. SC., MEC. E. (ALTA.)

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE
OF MASTER OF SCIENCE

MECHANICAL ENGINEERING

EDMONTON, ALBERTA

FALL 1982

THE UNIVERSITY OF ALBERTA
FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled NUMERICAL SOLUTION OF SOME SECOND-ORDER DIFFERENTIAL EQUATIONS submitted by BERNHARD MICHAEL JATZECK, B. SC., MEC. E. (ALTA.) in partial fulfilment of the requirements for the degree of MASTER OF SCIENCE.

ABSTRACT

The solution of some second-order differential equations using the method of piecewise linearization is examined. These are the undamped and damped hard spring equations (linear and nonlinear damping) and the Van der Pol and Mathieu equations.

Two methods of piecewise linearization are considered: solution by chords and solution by tangents. The two sets of results that are obtained are compared to the approximation to the exact solution of the undamped hard spring equation. The former method of linearization is chosen as the solution to the remainder of the equations that are examined.

Generally, the piecewise linearization solution gives a good agreement when compared to either the Runge-Kutta solution or the approximation to the exact solution (if available). Exceptions to this appear in some of the results for the Van der Pol and Mathieu equations.

The programmed piecewise linear methods are generally slower in execution time than either the Runge-Kutta solution or the approximation to the exact solution. The exceptions to this are the undamped hard spring equation and some instances for the Mathieu equation.

ACKNOWLEDGEMENTS

Several individuals contributed to the completion of this thesis. The following names I mention are among those whose efforts are deserving of my special thanks.

My supervisor, Dr. J. B. Haddow, was most helpful in the research and the writing. His guidance and advice in the material that was examined and the methods of solution that were used is greatly appreciated. I also wish to thank the other members of the examining committee for their efforts.

The personnel of Telsec Business Centre, Saskatoon, particularly Mrs. Carole Fontaine, were most patient in the preparation of the manuscript. I also wish to thank Mrs. Faye Magera, also of Saskatoon, for her initial efforts at this task.

The computer programming was a major endeavour and the advice of fellow graduate student Mr. Roger Toogood was a valuable asset. The advice of the consultants at the University of Alberta Computing Services, particularly Dr. Ron Torgerson, was also appreciated.

I am grateful to those department faculty and staff members for their efforts which allowed the completion of my degree. In particular, I thank Dr. D. G. Bellow, department chairman, for making it all possible.

A brief word of thanks also goes to some fellow students and some of the staff at the University of British Columbia, Department of Mechanical Engineering, from whom I learned much while I was there.

Finally, I wish to thank the many authors and/or publishers who were kind to allow me to use or cite material from their works.

TABLE OF CONTENTS

<u>Chapter</u>	<u>Page</u>
ABSTRACT	iv
ACKNOWLEDGEMENTS	v
LIST OF TABLES	viii
LIST OF FIGURES.....	ix
LIST OF SYMBOLS	xv
I. INTRODUCTION	1
II. HARD SPRING EQUATION (UNDAMPED)	5
A. Preliminary Comments	5
B. Approximation to Exact Solution	5
C. Derivation of Piecewise Linear Solution by Chords	9
D. Derivation of Piecewise Linear Solution by Tangents	16
E. Results	23
III. HARD SPRING EQUATION (LINEAR DAMPING)	35
A. Preliminary Comments	35
B. Derivation of Piecewise Linear Solution	37
C. Runge-Kutta Solution	39
D. Results	39
IV. HARD SPRING EQUATION (NONLINEAR DAMPING)	58
A. Preliminary Comments	58
B. Derivation of Piecewise Linear Solution	58
C. Results	62
V. VAN DER POL EQUATION	79
A. Preliminary Comments	79
B. Derivation of Piecewise Linear Solution	80
C. Results	87

TABLE OF CONTENTS (continued)

<u>Chapter</u>	<u>Page</u>
VI. MATHIEU EQUATION	135
A. Preliminary Comments	135
B. Generation of Characteristic Number	135
C. Series Approximation to Exact Solution	142
D. Derivation of Piecewise Linear Solution	147
E. Results	150
VII. DISCUSSION	184
VIII. CONCLUSIONS	197
IX. RECOMMENDATIONS FOR FURTHER WORK	199
A. Mathieu's Equation	199
B. Two-Degree-of-Freedom Nonlinear Equations	200
REFERENCES	201
APPENDIX	206
A. HSPRING	207
B. HSPRINGC	214
C. HSPRINGQ	221
D. VANDERPOL	228
E. MATHIEU	237
VITA	247

LIST OF TABLES

<u>Table</u>		<u>Page</u>
2.1	Undamped Hard Spring Equation Periods for $a = 1.0$, $b = 0.15$	24
2.2	Undamped Hard Spring Equation Periods for $a = 1.0$, $b = 2.0$	26

LIST OF FIGURES

<u>Figure</u>		<u>Page</u>
1	Methods of Piecewise Linearization	4
2.1	Mass-Spring System (Undamped)	6
2.2	Approximation by Chords	9
2.3	Chord Segment Construction	12
2.4	Approximation by Tangents	17
2.5	Tangent Segment Construction	18
2.6	Solutions to Hard Spring Equation (Undamped) for $a = 1.0$, $b = 0.15$, $\Delta t = \tau_{\pi/4}/100.0$, $\Delta x = 0.20$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	27
2.7	Solutions to Hard Spring Equation (Undamped) for $a = 1.0$, $b = 0.15$, $\Delta t = \tau_{\pi/4}/100.0$, $\Delta x = 0.10$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	28
2.8	Solutions to Hard Spring Equation (Undamped) for $a = 1.0$, $b = 2.0$, $\Delta t = \tau_{\pi/4}/100.0$, $\Delta x = 0.20$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	29
2.9	Solutions to Hard Spring Equation (Undamped) for $a = 1.0$, $b = 2.0$, $\Delta t = \tau_{\pi/4}/100.0$, $\Delta x = 0.10$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	30
2.10	Execution Times for Solutions to Hard Spring Equation (Undamped) for $a = 1.0$, $b = 0.15$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	32
2.11	Execution Times for Solutions to Hard Spring Equation (Undamped) for $a = 1.0$, $b = 2.0$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	33
3.1	Mass-Spring System (Linear Damping)	36
3.2	Solutions to Hard Spring Equation (Linear Damping) for $a = 1.0$, $b = 0.15$, $\frac{c}{M} = 0.0$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	41
3.3	Solutions to Hard Spring Equation (Linear Damping) for $a = 1.0$, $b = 2.0$, $\frac{c}{M} = 0.0$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	43
3.4	Solutions to Hard Spring Equation (Linear Damping) for $a = 1.0$, $b = 0.15$, $\frac{c}{M} = 0.10$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	45

LIST OF FIGURES (cont'd)

<u>Figure</u>		<u>Page</u>
3.5	Solutions to Hard Spring Equation (Linear Damping) for $a = 1.0$, $b = 2.0$, $\frac{c}{M} = 0.10$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	47
3.6	Solutions to Hard Spring Equation (Linear Damping) for $a = 1.0$, $b = 0.15$, $\frac{c}{M} = 0.25$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	49
3.7	Solutions to Hard Spring Equation (Linear Damping) for $a = 1.0$, $b = 2.0$, $\frac{c}{M} = 0.25$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	51
3.8	Execution Times for Solutions to Hard Spring Equation (Linear Damping) for $a = 1.0$, $\frac{c}{M} = 0.0$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	53
3.9	Execution Times for Solutions to Hard Spring Equation (Linear Damping) for $a = 1.0$, $\frac{c}{M} = 0.10$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	54
3.10	Execution Times for Solutions to Hard Spring Equation (Linear Damping) for $a = 1.0$, $\frac{c}{M} = 0.25$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	55
4.1	Mass-Spring System (Nonlinear Damping)	59
4.2	Solutions to Hard Spring Equation (Nonlinear Damping) for $a = 1.0$, $b = 0.15$, $\frac{q}{M} = 0.0$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	64
4.3	Solutions to Hard Spring Equation (Nonlinear Damping) for $a = 1.0$, $b = 2.0$, $\frac{q}{M} = 0.0$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	66
4.4	Solutions to Hard Spring Equation (Nonlinear Damping) for $a = 1.0$, $b = 0.15$, $\frac{q}{M} = 0.10$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	68
4.5	Solutions to Hard Spring Equation (Nonlinear Damping) for $a = 1.0$, $b = 2.0$, $\frac{q}{M} = 0.10$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	70
4.6	Solutions to Hard Spring Equation (Nonlinear Damping) for $a = 1.0$, $b = 0.15$, $\frac{q}{M} = 0.25$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	72

LIST OF FIGURES (cont'd)

<u>Figure</u>	<u>Page</u>
4.7 Solutions to Hard Spring Equation (Nonlinear Damping) for $a = 1.0$, $b = 2.0$, $\frac{q}{M} = 0.25$, $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	74
4.8 Execution Times for Solutions to Hard Spring Equation (Nonlinear Damping) for $a = 1.0$, $\frac{q}{M} = 0.0$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	75
4.9 Execution Times for Solutions to Hard Spring Equation (Nonlinear Damping) for $a = 1.0$, $\frac{q}{M} = 0.10$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	76
4.10 Execution Times for Solutions to Hard Spring Equation (Nonlinear Damping) for $a = 1.0$, $\frac{q}{M} = 0.25$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	77
5.1 Typical Interval Layout (Van der Pol Equation)	82
5.2 Layout of Peak or Trough (Van der Pol Equation)	84
5.3 Solutions to Van der Pol Equation for $\mu = 0.25$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	89
5.4 Phase Trajectories for Van der Pol Equation for $\mu = 0.25$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	91
5.5 Solutions to Van der Pol Equation for $\mu = 0.25$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.0$, $\dot{x}(0) = 0.0$	93
5.6 Phase Trajectories for Van der Pol Equation for $\mu = 0.25$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.0$, $\dot{x}(0) = 0.0$	95
5.7 Solutions to Van der Pol Equation for $\mu = 0.25$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	97
5.8 Phase Trajectories for Van der Pol Equation for $\mu = 0.25$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	99
5.9 Solutions to Van der Pol Equation for $\mu = 1.0$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	102
5.10 Phase Trajectories for Van der Pol Equation for $\mu = 1.0$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	104

LIST OF FIGURES (cont'd)

<u>Figure</u>		<u>Page</u>
5.11	Solutions to Van der Pol Equation for $\mu = 2.0$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	106
5.12	Phase Trajectories for Van der Pol Equation for $\mu = 2.0$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	108
5.13	Solutions to Van der Pol Equation for $\mu = 5.0$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	110
5.14	Phase Trajectories for Van der Pol Equation for $\mu = 5.0$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$	112
5.15	Solutions to Van der Pol Equation for $\mu = 5.0$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.0$, $\dot{x}(0) = 0.0$	114
5.16	Phase Trajectories for Van der Pol Equation for $\mu = 5.0$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.0$, $\dot{x}(0) = 0.0$	116
5.17	Solutions to Van der Pol Equation for $\mu = 5.0$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	118
5.18	Phase Trajectories for Van der Pol Equation for $\mu = 5.0$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	120
5.19	Solutions to Van der Pol Equation for $\mu = 5.0$, $\Delta t_P = 0.05$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	122
5.20	Phase Trajectories for Van der Pol Equation for $\mu = 5.0$, $\Delta t_P = 0.05$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	124
5.21	Solutions to Van der Pol Equation for $\mu = 5.0$, $\Delta t_P = 0.25$, $\Delta t_{RK} = 0.25$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	126
5.22	Phase Trajectories for Van der Pol Equation for $\mu = 5.0$, $\Delta t_P = 0.25$, $\Delta t_{RK} = 0.25$, $x(0) = 2.5$, $\dot{x}(0) = 0.0$	128
5.23	Execution Times for Van der Pol Equation Solutions for $\mu = 0.25$, $x(0) = 3.0$, $\dot{x}(0) = 0.0$	131
5.24	Execution Times for Van der Pol Equation Solutions for $\mu = 1.0$, $x(0) = 3.0$, $\dot{x}(0) = 0.0$	132
5.25	Execution Times for Van der Pol Equation Solutions for $\mu = 2.0$, $x(0) = 3.0$, $\dot{x}(0) = 0.0$	133

LIST OF FIGURES (cont'd)

<u>Figure</u>		<u>Page</u>
5.26	Execution Times for Van der Pol Equation Solutions for $\mu = 5.0$, $x(0) = 3.0$, $\dot{x}(0) = 0.0$	134
6.1	Typical Interval Layout (Mathieu Equation)	149
6.2	Solutions to Mathieu Equation for $a_{\text{GUESS}} = -50.0$, $q = 1.0$, $\Delta z = \pi/200.0$, $y(0) = 0.0$, $y'(0) = 1.0$	153
6.3	Solutions to Mathieu Equation for $a_{\text{GUESS}} = 10.0$, $q = 1.0$, $\Delta z = \pi/200.0$, $y(0) = 1.0$, $y'(0) = 0.0$	155
6.4	Solutions to Mathieu Equation for $a_{\text{GUESS}} = 30.0$, $q = 1.0$, $\Delta z = \pi/200.0$, $y(0) = 0.0$, $y'(0) = 1.0$	157
6.5	Solutions to Mathieu Equation for $a_{\text{GUESS}} = 10.0$, $q = 10.0$, $\Delta z = \pi/200.0$, $y(0) = 0.0$, $y'(0) = 1.0$	159
6.6	Solutions to Mathieu Equation for $a_{\text{GUESS}} = 30.0$, $q = 10.0$, $\Delta z = \pi/200.0$, $y(0) = 0.0$, $y'(0) = 1.0$	161
6.7	Solutions to Mathieu Equation for $a_{\text{GUESS}} = -50.0$, $q = 10.0$, $\Delta z = \pi/200.0$, $y(0) = 0.0$, $y'(0) = 1.0$	163
6.8	Solutions to Mathieu Equation for $a_{\text{GUESS}} = 10.0$, $q = 20.0$, $\Delta z = \pi/200.0$, $y(0) = 1.0$, $y'(0) = 0.0$	165
6.9	Solutions to Mathieu Equation for $a_{\text{GUESS}} = -50.0$, $q = 20.0$, $\Delta z = \pi/200.0$, $y(0) = 0.0$, $y'(0) = 1.0$	167
6.10	Solutions to Mathieu Equation for $a_{\text{GUESS}} = 30.0$, $q = 30.0$, $\Delta z = \pi/200.0$, $y(0) = 0.0$, $y'(0) = 1.0$	169
6.11	Solutions to Mathieu Equation for $a_{\text{GUESS}} = -50.0$, $q = 30.0$, $\Delta z = \pi/200.0$, $y(0) = 0.0$, $y'(0) = 1.0$	171
6.12	Solutions to Mathieu Equation for $a_{\text{GUESS}} = 30.0$, $q = 40.0$, $\Delta z = \pi/200.0$, $y(0) = 1.0$, $y'(0) = 0.0$	173
6.13	Solutions to Mathieu Equation for $a_{\text{GUESS}} = -50.0$, $q = 40.0$, $\Delta z = \pi/200.0$, $y(0) = 1.0$, $y'(0) = 0.0$	175
6.14	Solutions to Mathieu Equation for $a_{\text{GUESS}} = -50.0$, $q = 40.0$, $\Delta z = \pi/400.0$, $y(0) = 1.0$, $y'(0) = 0.0$	178
6.15	Execution Times for Solutions to Mathieu Equation for $a_{\text{GUESS}} = 10.0$, $q = 1.0$, $y(0) = 1.0$, $y'(0) = 0.0$	180

LIST OF FIGURES (cont'd)

<u>Figure</u>		<u>Page</u>
6.16	Execution Times for Solutions to Mathieu Equation for $a_{\text{GUESS}} = 30.0$, $q = 20.0$, $y(0) = 1.0$, $y'(0) = 0.0$	181
6.17	Execution Times for Solutions to Mathieu Equation for $a_{\text{GUESS}} = -50.0$, $q = 40.0$, $y(0) = 1.0$, $y'(0) = 0.0$	182
7.1	Solutions to Mathieu Equation for $a_{\text{GUESS}} = -50.0$, $q = 20.0$, $\Delta z = \pi/200.0$, $y(0) = 0.0$, $y'(0) = 1.0$	193
7.2	Stability Chart for Piecewise Linear Solution to Mathieu Equation	194

LIST OF SYMBOLS

a, b	Hard spring restoring force coefficients
a	Characteristic number (general use and for non-zero initial displacement and zero initial velocity)
a_{GUESS}	Initial guess at characteristic number
a_{IN}	Input value for characteristic number continued fraction (scan phase)
a_{OUT}	Output for preceding continued fraction (scan phase)
a_{MID}	Input value for characteristic number continued fraction (interval halving search)
a_{MIDOUT}	Output value for preceding continued fraction
$a_{\text{UP}}, a_{\text{LOW}}$	Upper and lower bounds, respectively (scan phase)
a_{2n}	Characteristic number (even order, non-zero initial displacement, and zero initial velocity)
A_{2r}	Coefficient for series approximation to Mathieu function (even order and preceding initial conditions)
a_{2n+1}	Characteristic number (odd order and preceding initial conditions)
A_{2r+1}	Coefficient for series approximation to Mathieu function (odd order and preceding initial conditions)
b_{2n+1}	Characteristic number (odd order, zero initial displacement and non-zero initial velocity)
B_{2n+1}	Coefficient for series approximation to Mathieu function (odd order and preceding initial conditions)
b_{2n+2}	Characteristic number (even order and preceding initial conditions)
B_{2r+2}	Coefficient for series approximation to Mathieu function (even order and preceding initial conditions)
A, B	General equation coefficients for linear segment solution (trigonometric)
c, p	Damping constant and angular frequency, respectively, for linear approximation
c	Damping constant

LIST OF SYMBOLS (cont'd)

C, D	General equation coefficients for $\ddot{x}[(\Delta t)_i]$ (hyperbolic)
$f(a, q, n)$	General term for characteristic number continued fraction
$f(x, \dot{x}, t)$	Generalized term representing damping and restoring forces
F	Generalized restoring force
F_0	Restoring force at beginning of segment (method of chords)
F_1	Restoring force at end of segment (method of chords)
F'_0	Slope of tangent line at x_0 (method of tangents)
F'_1	Slope of tangent line at x_1 (method of tangents)
$F(\lambda, \phi)$	Incomplete elliptic integral of first kind
k_0	Slope of linear segment (method of chords)
$K(\lambda^2)$	Complete elliptic integral of first kind
m	Number of linear segments per quarter-cycle (piecewise linear solution)
M	System mass
n	Number of increments per quarter-cycle (approximation to exact solution of undamped hard spring equation)
n	$c/2M$
p	Trigonometric angular frequency for linear segment (piecewise linear solution)
p^*	Damped natural frequency
p^{**}	$\sqrt{n^2 - p^2}$ (Van der Pol equation)
p_m	Trigonometric angular frequency for mth segment
q	Nonlinear damping constant (hard spring equation, nonlinear damping)
q	Parameter for a (Mathieu equation)
s	ip

LIST OF SYMBOLS (cont'd)

s_m	Hyperbolic angular frequency for mth segment (Mathieu equation)
t	Time
t_0	Cumulative time at beginning of linear segment (method of chords)
t_1	Cumulative time at end of linear segment (method of chords)
$(t_1 - t_0)_i$	Interim value for $t_1 - t_0$ for Newton-Raphson relations (damped hard spring equation)
t_{C0}	Cumulative time at beginning of linear segment (method of tangents)
t_{C1}	Cumulative time at end of linear segment (method of tangents)
x	Displacement
$x[(t_1 - t_0)_i]$	Displacement using $(t_1 - t_0)_i$ (damped hard spring equation)
\dot{x}	Velocity
$\dot{x}[(t_1 - t_0)_i]$	Velocity using $(t_1 - t_0)_i$ (damped hard spring equation)
$\dot{x}[(\Delta t)_i]$	Velocity using $(\Delta t)_i$
\ddot{x}	Acceleration
$\ddot{x}[(t_1 - t_0)_i]$	Acceleration using $(t_1 - t_0)_i$ (damped hard spring equation)
$\ddot{x}[(\Delta t)_i]$	Acceleration using $(\Delta t)_i$
x_0	Displacement at beginning of linear segment
x_1	Displacement at end of linear segment
\dot{x}_0	Velocity at beginning of linear segment (method of chords)
\dot{x}_1	Velocity at end of linear segment (method of chords)
x_0^*	Relative equilibrium point for linear segment (method of chords)

LIST OF SYMBOLS (cont'd)

x_F	Estimated value for x_{11}
x_{PREV}	Previous value of estimated linear segment displacement
x_{C0}	Crossing point tangents from x_0 to x_1 for (method of tangents)
x_{C1}	Crossing point for tangents from x_1 and following linear segment (method of tangents)
x_{11}	Displacement at peak or trough
x_{12}	Displacement at end of linear segment in which peak or trough occurs
\dot{x}_{AV}	Average segment velocity (hard spring equation, nonlinear damping)
\dot{x}_{C0}	Velocity at x_{C0} (method of tangents)
\dot{x}_{C1}	Velocity at x_{C1} (method of tangents)
x_0	Initial system displacement
y	Displacement (Mathieu equation)
y'	Velocity (Mathieu equation)
y	Acceleration (Mathieu equation)
y_{m-1}, y_m	Displacements at end of $(m - 1)$ th and m th segments, respectively
y'_{m-1}, y'_m	Velocities at end of $(m - 1)$ th and m th segments, respectively
y_0	Displacement at beginning of segment
y'_0	Velocity at beginning of segment
Y_0	Initial system displacement (Mathieu equation)
Y'_0	Initial velocity (Mathieu equation)
z	Independent variable (Mathieu equation)
α, β	Coefficients for approximation to exact solution to Mathieu equation
α, β, γ	Trigonometric equation coefficients for finding $t_1 - t_0$ (undamped hard spring equation)

LIST OF SYMBOLS (cont'd)

α_v	Generalized numerator for continued fraction
β_v	Generalized denominator for continued fraction
γ_{n+1}, γ_n	Sum of characteristic number continued fraction terms prior to $(n + 1)$ th or n th terms
Δt	Time increment (approximation to exact solution to undamped hard spring equation)
Δt_p	Time interval for piecewise linear solution (Van der Pol equation)
Δt_{RK}	Time interval for Runge-Kutta solution (Van der Pol equation)
Δt_1	Interval from t_0 to time at which peak or trough occurs
Δt_2	Interval from peak or trough to t_1
$(\Delta t)_1$	Interim value of Δt_1 as used in Newton-Raphson method
Δx	Displacement increment (piecewise linear solution
Δz	Increment of z
θ	Solution to preceding trigonometric equation
λ^2	Parameter for $F(\lambda, \phi)$
μ	Equation parameter (Van der Pol equation)
v	Term number for continued fraction
τ	Period of oscillation (Mathieu equation)
τ_π	Period of oscillation (approximation to exact solution of undamped hard spring equation)
$\tau_\pi/4$	Quarter-period of oscillation (approximation to exact solution of undamped hard spring equation)
ϕ	Amplitude for $F(\lambda, \phi)$
$\phi(r)$	Order of approximation to exact solution of Mathieu equation
ϕ_{n+1}	Denominator for characteristic number continued fraction

LIST OF SYMBOLS (cont'd)

ω Angular frequency of oscillation for Jacobian elliptic function

I. INTRODUCTION

Oscillating systems are common in mechanical and electrical engineering, as well as in other areas of science such as chemistry and biology. These can often be described by using differential equations, either nonlinear or linear with time-varying coefficients.

Exact solutions for these equations are often difficult to obtain (if at all), but reasonably good approximations can usually be achieved using a variety of methods. Some of the better-known ones are variation of parameters [1], reversion [2], and perturbation [3].

One method that is mentioned in some textbooks but is often not given as much prominence as the others is that of piecewise linearization, the approximation of an equation by one that is linear. This approach is investigated in this thesis, applying it to a number of second-order differential equations, comparing the results with either an approximation to an exact solution (if available), such as [4], or to a solution obtained from a Runge-Kutta method [5].

What this method does is to approximate the equation being examined:

$$\ddot{x} + f(x, \dot{x}, t) = 0, \quad (1.1)$$

by one that is linear. The terms corresponding to the damping and restoring forces in (1.1) are represented by $f(x, \dot{x}, t)$. The linear equation is that for a mass-spring oscillator:

$$\left. \begin{aligned} \ddot{x} + \frac{c}{M} \dot{x} + \frac{k}{M} x &= 0 \\ &= \ddot{x} + 2n\dot{x} + p^2 x, \end{aligned} \right\} \quad (1.2)$$

where:

c = Coefficient of damping,

M = System mass,

k = Spring constant,

p = Undamped natural frequency.

The key is to approximate the coefficients of x and \dot{x} in (1.1) by constants that would correspond to n and p in (1.2).

This is done by considering the restoring force-displacement curve of (1.1). For a small section of this curve, a straight line could be used as an approximation to it over the same region in which this section lies [6]. The curve can be thought of consisting of a sequence of these small sections, and each will have its own unique linear approximation.

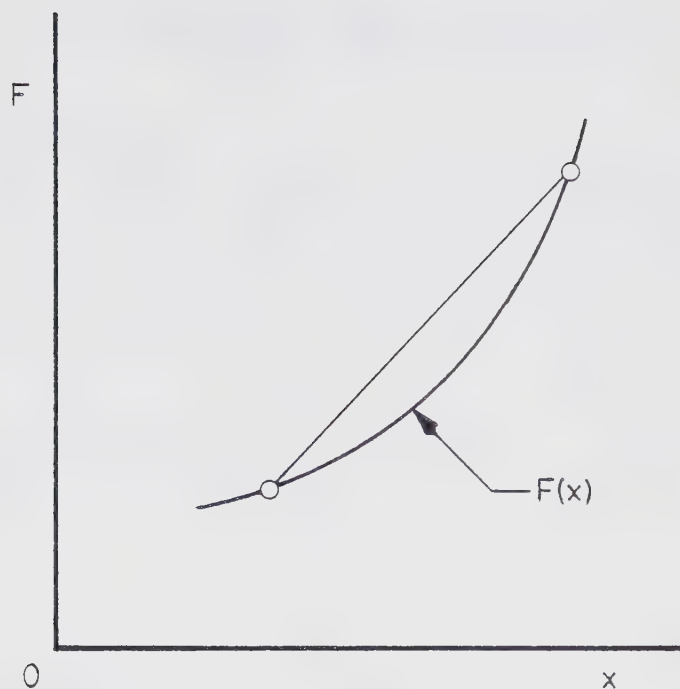
The approximation of (1.1) by (1.2) will only be effective over a small interval of displacement or time. The initial and final displacements of this interval will determine the length of the straight line approximation of the restoring force-displacement curve just mentioned. The slope of this line will be k in (1.2). If the damping expression in (1.1) is nonlinear, c would have to be found by iteration over the interval of displacement or time, depending upon the nature of the nonlinearity. The solution to (1.2) can now be obtained, using the initial conditions for the interval in question. The end conditions of this interval become the initial values of the one following, and the procedure is repeated with new values of c and k , as well as n and p , being determined.

By successfully solving (1.2) for each interval, an approximate

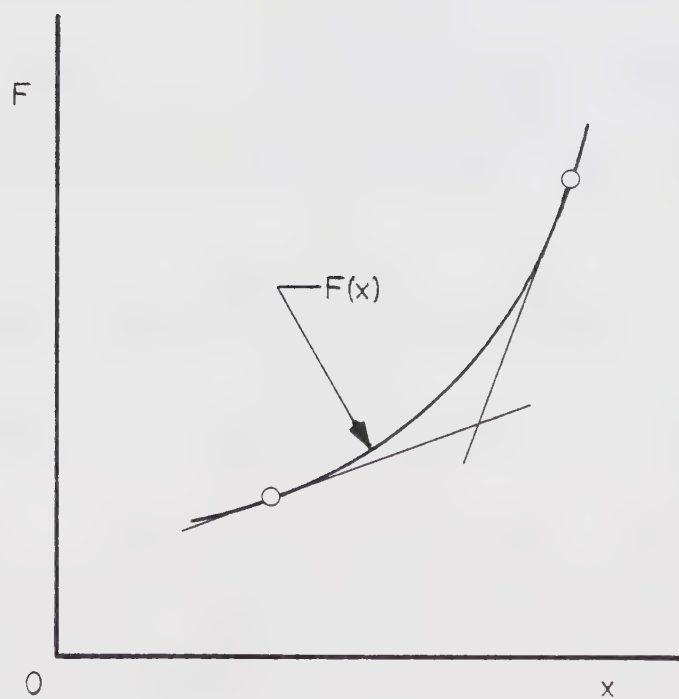
solution for (1.1) can be obtained, approaching the actual one as the intervals used become smaller.

The straight line approximation of the restoring force-displacement curve is accomplished by running a chord between the two endpoints of the section (Figure 1A) or tangents at the respective locations (Figure 1B). These points are located on the curve by knowing the initial and final displacements of the interval over which (1.2) is to be used. This force is shown in the figure as F and the displacement as x .

With this in mind, one can now apply this method to specific problems, as shall be seen in the following chapters.



A. CHORDS



B. TANGENTS

FIGURE 1 - METHODS OF PIECEWISE LINEARIZATION

II. HARD SPRING EQUATION (UNDAMPED)

A. Preliminary Comments

One of the simplest examples of the application of piecewise linearization is that of the hard undamped spring, the equation of which is:

$$\ddot{x} + ax + bx^3 = 0, \quad (2.1.1)$$

where:

$$a, b > 0. \quad (2.1.2)$$

See Figure 2.1 for details [7]. This is considered to be the equation for a hard spring since the restoring force increases quicker with respect to deflection than if it were linear [8]. The overall time span to be considered is the first quarter-cycle, since (2.1.1) is an odd function.

The following initial conditions will be used:

$$x(0) = x_0, \quad \dot{x}(0) = 0. \quad (2.1.3)$$

The cases examined had values for a and b of 1.0 and 0.15 respectively, and 1.0 and 2.0, respectively, to see how the solutions behaved with a low and a high degree of nonlinearity.

B. Approximation to Exact Solution

An exact solution for this equation exists, making use of Jacobian elliptic functions. The derivations of this result can be found in any

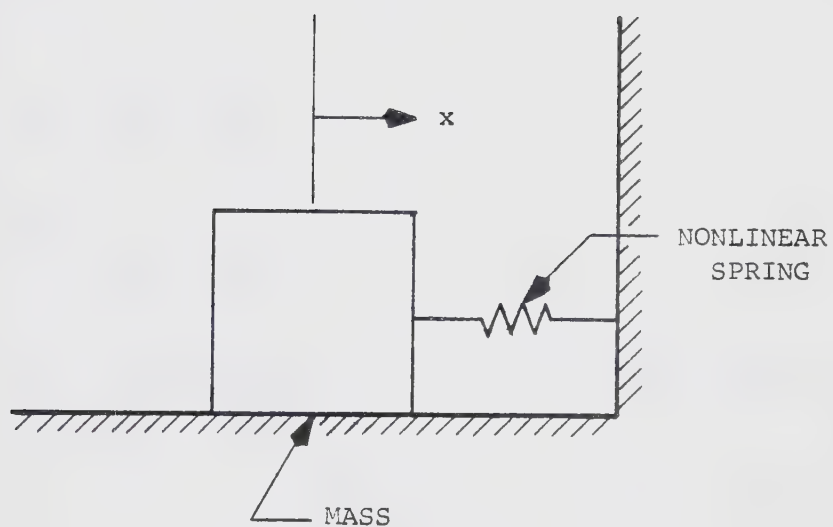


FIGURE 2.1 - MASS-SPRING SYSTEM (UNDAMPED) [7]

one of McLachlan [9], Cunningham [10], or Soudack [11]. This exact solution starts with:

$$\begin{aligned}
 t &= \frac{1}{(a + bX_0^2)^{1/2}} \int_0^\phi \frac{d\psi}{(1 - \lambda^2 \sin^2 \psi)^{1/2}} \\
 &= \frac{F(\lambda, \phi)}{(a + bX_0^2)^{1/2}}, \quad (2.2.1)
 \end{aligned}$$

where:

$$\begin{aligned}
 \phi &= \cos^{-1} \left(\frac{x}{X_0} \right), \\
 \lambda^2 &= \frac{bX_0^2}{2(a + bX_0^2)},
 \end{aligned}$$

$F(\lambda, \phi)$ = Incomplete elliptical integral of the first kind.

By applying the assumptions (2.1.2) and initial conditions (2.1.3), the solution to (2.1.1) is, together with (2.2.1)[12], [13]:

$$x = X_0 \operatorname{cn}(\lambda, \omega t), \quad (2.2.2)$$

where:

$$\omega^2 = (a + bX_0^2). \quad (2.2.3)$$

At the end of the first quarter-cycle, ϕ is $\pi/2$. Using the definition for some of the terms of (2.1.4), $F(\lambda, \phi)$ is known as the

complete elliptic integral of the first kind, which is given the designation of $K(\lambda^2)$ [14]. But, it is not always practical to refer to tables each time one varies a and/or b , and so an easier means of obtaining this value should be used. An infinite series does exist for this and for a specific value for λ^2 , one can use the following [15]:

$$\begin{aligned}
 K(\lambda^2) = \frac{\pi}{2} \left[1 + \left(\frac{1}{2} \right)^2 (\lambda^2) + \left(\frac{1 \cdot 3}{2 \cdot 4} \right)^2 (\lambda^2)^3 + \dots \right], \\
 (|\lambda^2| < 1),
 \end{aligned}
 \tag{2.2.4}$$

substituting this for $F(\lambda, \phi)$ in (2.2.1)[9], [10], [11] for calculating the period, and combining this with the definition for λ^2 to generate the solution. Since only the first 100 terms of (2.2.4) will be calculated, the solution obtained, which will be used as a benchmark for the piecewise linear results, will only be an approximation to (2.2.2) [12], [13] and shall be subsequently be referred to as such.

A method to generate the values for $\text{cn}(\lambda, \omega t)$ is required. One does exist, making use of what is known as the "arithmetic-geometric mean" approach [16]. The computer program that was used for solving (2.1.1) accessed a subroutine library devised by the computing centre at the University of British Columbia which uses this method [4].

From McLachlan [17], the period of oscillation is:

$$\tau_{\pi} = \frac{4 K(\lambda^2)}{(a + b x_0^2)^{1/2}}
 \tag{2.2.5}$$

and so, for the quarter-cycle,

$$\tau_{\pi/4} = \frac{K(\lambda^2)}{(a + bX_0^2)^{1/2}} .$$

It should be noted that it is necessary to know the solution period as a basis for comparing the two piecewise linear methods that are to follow.

The actual solutions obtained by these methods are to be compared as well as with the one obtained by this approximation to the exact solution. The curve for the approximation to (2.2.2)[4], [15], [17] is obtained by dividing the quarter-cycle into n increments of time (where n is an integer) and calculating the displacement by incrementing the value of t by Δt , which is defined as follows:

$$\Delta t = \frac{K(\lambda^2)}{n(a + bX_0^2)^{1/2}} .$$

C. Derivation of Piecewise Linear Solution by Chords

The following method is based on Timoshenko, et. al. [18].

Assume that a restoring force-displacement curve for the system shown in Figure 2.1 can be approximated by a series of linear segments, as shown in Figure 2.2. The force is designated as F , while displacement is x . Assume further that the initial total displacement is divided into m equal sections of length Δx (m being an integer), with the total displacement being X_0 , or:

$$\Delta x = \frac{X_0}{m} .$$

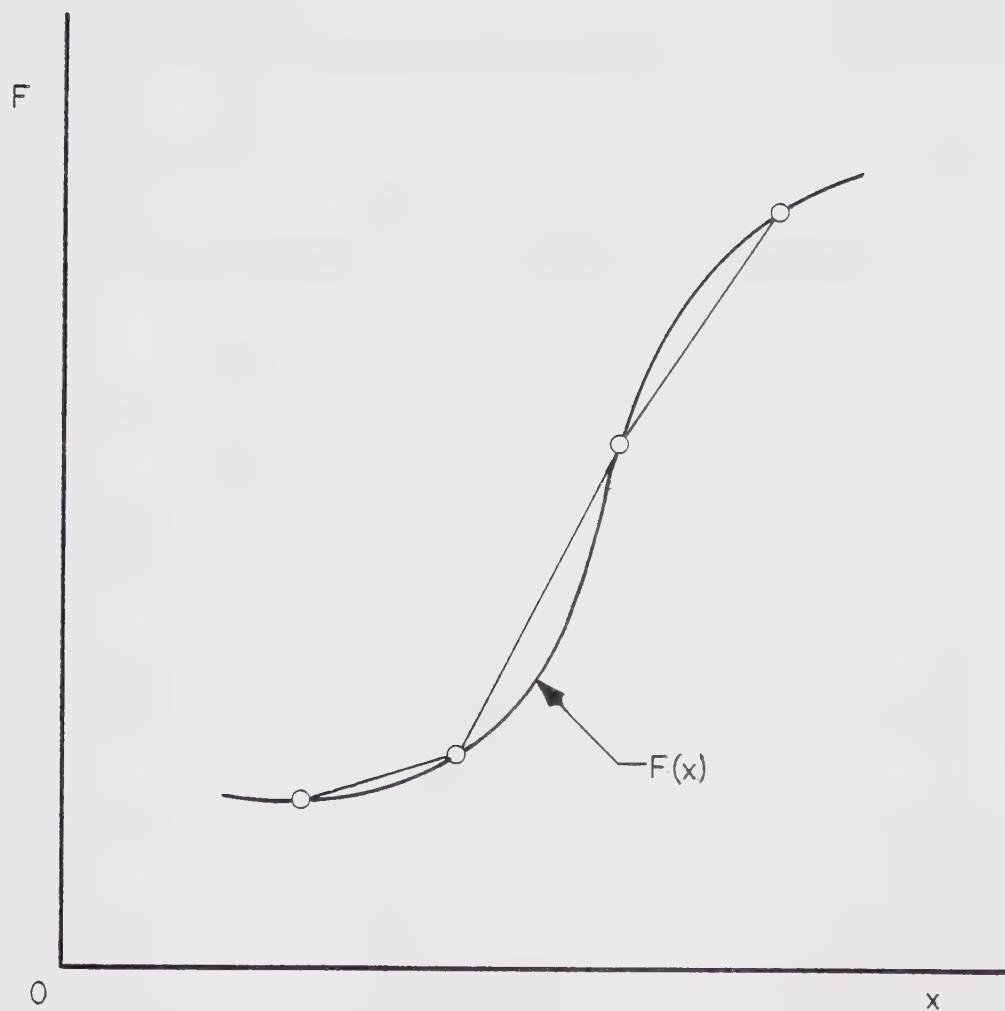


FIGURE 2.2 - APPROXIMATION BY CHORDS

Now consider, specifically for (2.1.1), two adjacent points (A and B) with displacements x_0 and x_1 , respectively, and times t_0 and t_1 , respectively (Figure 2.3). The system is initially at rest with displacement X_0 , and then released. For this:

x_0 = Initial displacement of segment

$$= x(t_0),$$

x_1 = Final displacement of segment

$$= x(t_1),$$

k_0 = Slope of segment,

F_0 = Restoring force at beginning of segment

$$= M(ax_0 + bx_0^3),$$

F_1 = Restoring force and end of segment

$$= M(ax_1 + bx_1^3),$$

x_0^* = x - axis crossing point for chord,

M = System mass.

It can be also seen that:

$$\Delta x = x_0 - x_1$$

and:

$$\begin{aligned} \frac{k_0}{M} &= \frac{F_0 - F_1}{M\Delta x} \\ &= \frac{a(x_0 - x_1) + b(x_0^3 - x_1^3)}{x_0 - x_1} . \end{aligned}$$

The equation of motion for line AB is (for $x_1 \leq x \leq x_0$,

$t_1 \geq t \geq t_0$) [19]:

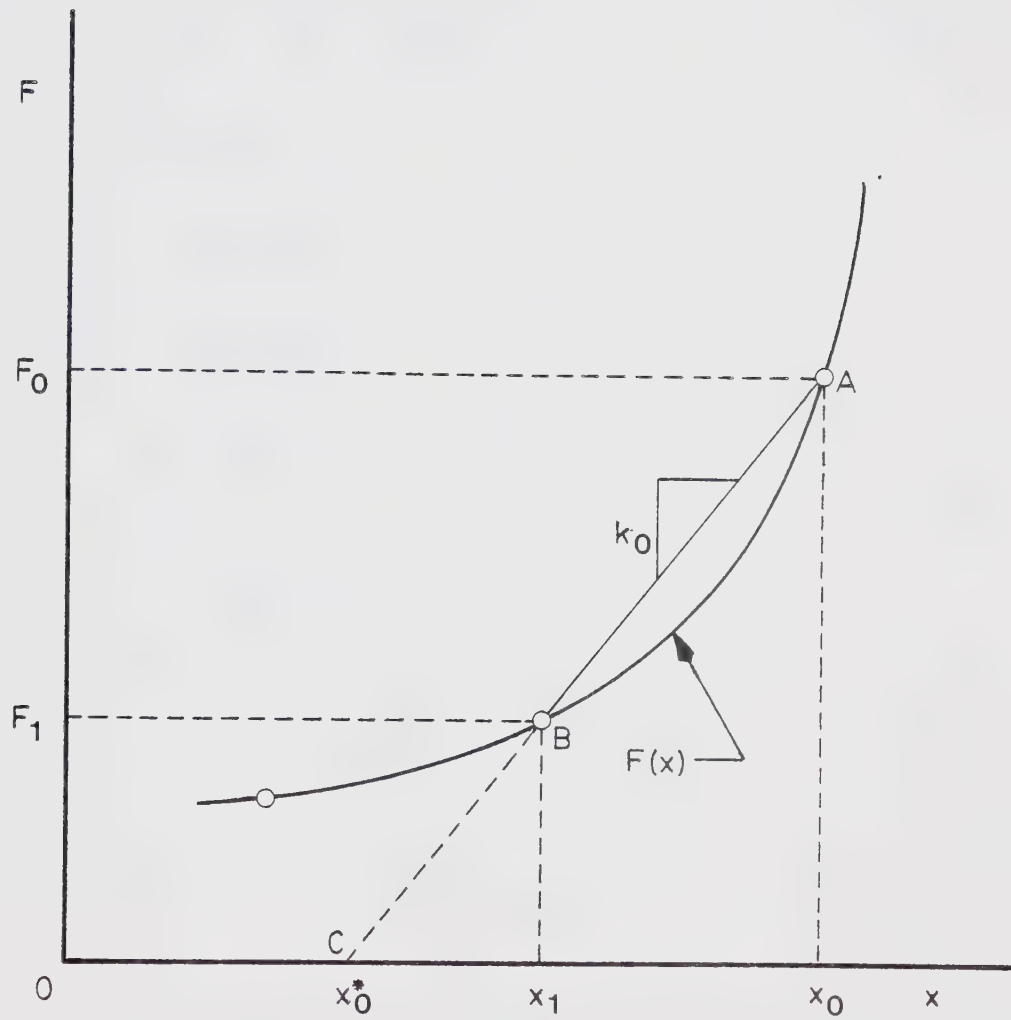


FIGURE 2.3 - CHORD SEGMENT CONSTRUCTION

$$\frac{d^2}{dt^2} (x - x_0^*) + \frac{k_0}{M} (x - x_0^*) = 0. \quad (2.3.1)$$

Referring to Figure 2.3, the value of x_0^* is determined as follows. AB is extended to the displacement axis (line AC) and the ratio of BC to AC is found. This is done by using the point-of-division methods from plane analytic geometry [20]:

$$\begin{aligned} r &= \frac{BC}{AC} \\ &= \frac{F_1 - 0}{F_0 - 0} \\ &= \frac{x_1 - x_0^*}{x_0 - x_0^*}, \\ (x_0 - x_0^*)r &= x_1 - x_0^*, \\ x_0^*(1 - r) &= x_1 - x_0 r, \\ x_0^* &= \frac{x_1 - x_0 r}{1 - r} \\ &= \left(\frac{1}{1 - F_1/F_0} \right) [x_1 - (F_1/F_0)x_0]. \end{aligned}$$

The solution to (2.2.1) is of the form [21]:

$$x = x_0^* + A \cos p(t - t_0) + B \sin p(t - t_0),$$

where:

$$p = \sqrt{\frac{k_0}{M}}.$$

Differentiating gives the velocity [21]:

$$\dot{x} = -pA \sin p(t - t_0) + pB \cos p(t - t_0).$$

Consider what happens at the beginning of the segment. The results are:

$$x_0 - x_0^* = A \cos p(0) + B \sin p(0) ,$$

$$A = x_0 - x_0^* ,$$

$$\dot{x}_0 = -pA \sin p(0) + pB \cos p(0) ,$$

$$B = \frac{\dot{x}_0}{p} .$$

Therefore, the general solution for any segment would be [21]:

$$\begin{aligned} x = & x_0^* + (x_0 - x_0^*) \cos p(t - t_0) \\ & + \frac{\dot{x}_0}{p} \sin p(t - t_0) , \end{aligned} \quad (2.3.2)$$

with the velocity [21]:

$$\begin{aligned} \dot{x} = & -p(x_0 - x_0^*) \sin p(t - t_0) \\ & + \dot{x}_0 \cos p(t - t_0) . \end{aligned} \quad (2.3.3)$$

For any given x_0 , the value of t_0 is the total time taken to traverse the preceding individual segments. The question now is what, for any given segment, this time would be. From (2.3.2), it can be seen that for a given segment, the time taken to go from x_0 to x_1 would be $(t_1 - t_0)$, where, in this case, t_1 is the total time taken to reach x_1 . Making the substitution of t_1 for t in (2.3.2) yields the following:

$$x_1 = x_0^* + (x_0 - x_0^*) \cos p(t_1 - t_0) + \frac{\dot{x}_0}{p} \sin p(t_1 - t_0).$$

This takes the form of the trigonometric equation [22]:

$$\gamma = \alpha \cos \theta - \beta \sin \theta,$$

where:

$$\left. \begin{aligned} \alpha &= x_0 - x_0^* \\ &> 0, \\ \beta &= \left| \frac{\dot{x}_0}{p} \right| \\ &> 0, \\ \gamma &= x_1 - x_0^* . \end{aligned} \right\} \quad (2.3.4)$$

Two possible solutions are possible for this equation [22]:

$$\theta_1 = -\tan^{-1}\left(\frac{\beta}{\alpha}\right) - \cos^{-1}\left(\frac{\gamma}{\sqrt{\alpha^2 + \beta^2}}\right),$$

$$\theta_2 = -\tan^{-1}\left(\frac{\beta}{\alpha}\right) + \cos^{-1}\left(\frac{\gamma}{\sqrt{\alpha^2 + \beta^2}}\right).$$

From $\tan^{-1} \delta = -\tan^{-1}(-\delta)$ [23] and the fact that $\gamma > 0$, plus $\theta > 0$ at a point right after $t = 0$,

$$\begin{aligned}
 \theta &= p(t_1 - t_0) \\
 &= \tan^{-1}\left(-\frac{\beta}{\alpha}\right) + \cos^{-1}\left(\frac{\gamma}{\sqrt{\alpha^2 + \beta^2}}\right), \quad (2.3.5)
 \end{aligned}$$

$$t_1 = t_0 + \frac{\theta}{p}.$$

using the definitions of (2.3.4).

Since the various values would have to be found iteratively, the quarter-period for this method would be the final value of t_1 .

D. Derivation of Piecewise Linear Solution by Tangents

As for the method of chords, the reference for this section is Timoshenko, et. al. [18].

Referring to the oscillator shown in Figure 2.1, consider now the case of the restoring force-displacement curve being approximated by a series of linear segments, but using tangents instead of chords (Figure 2.4). The force is designated as F and displacement as x . Assume further that, as before, the length of the segment is Δx . For two consecutive segments, specifically for (2.1.1), the construction would look like Figure 2.5, with:

- x_0 = Displacement at beginning of segment,
- x_1 = Displacement at end of segment,
- x_0^* = x - axis crossing point for tangent,
- x_{C0} = Point where tangents from x_0 and x_1 cross
 $= x(t_{C0})$,
- x_{C1} = Point where tangent from x_1 crosses one
 from endpoint of next segment,
 $= x(t_{C1})$,

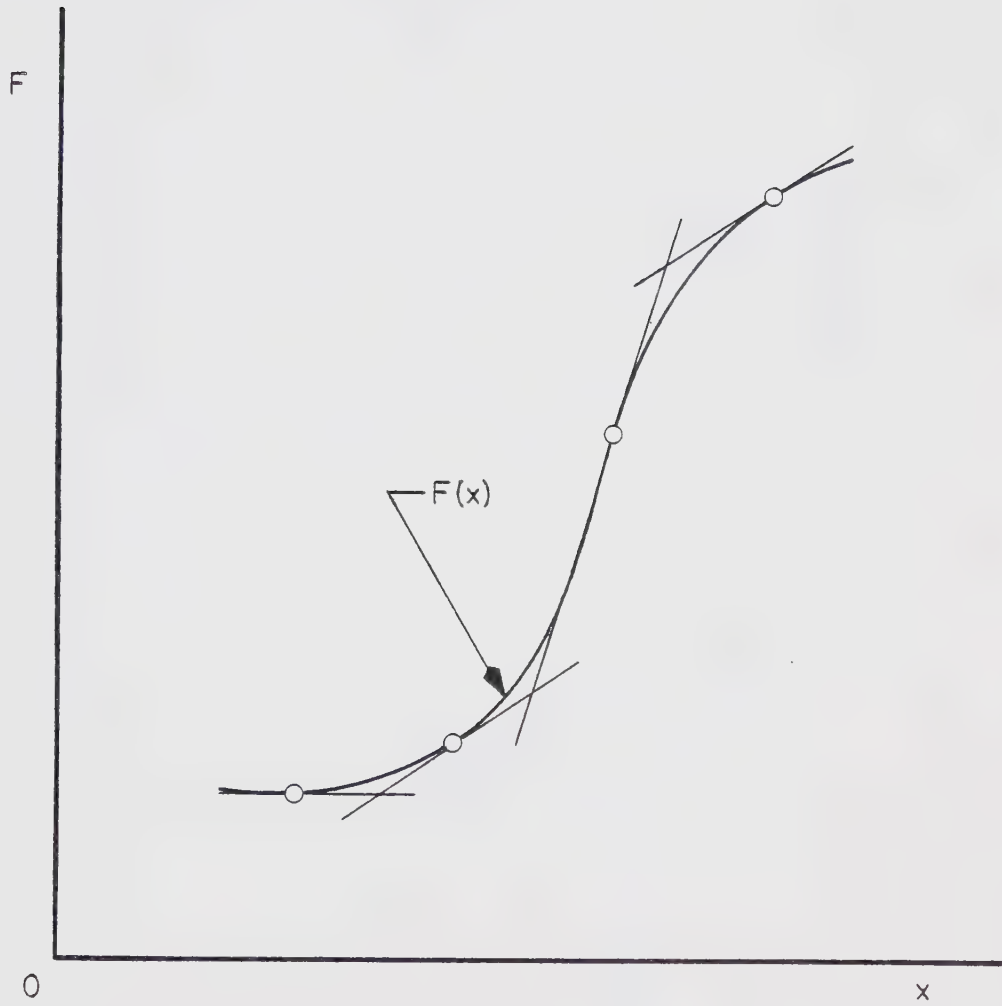


FIGURE 2.4 - APPROXIMATION BY TANGENTS

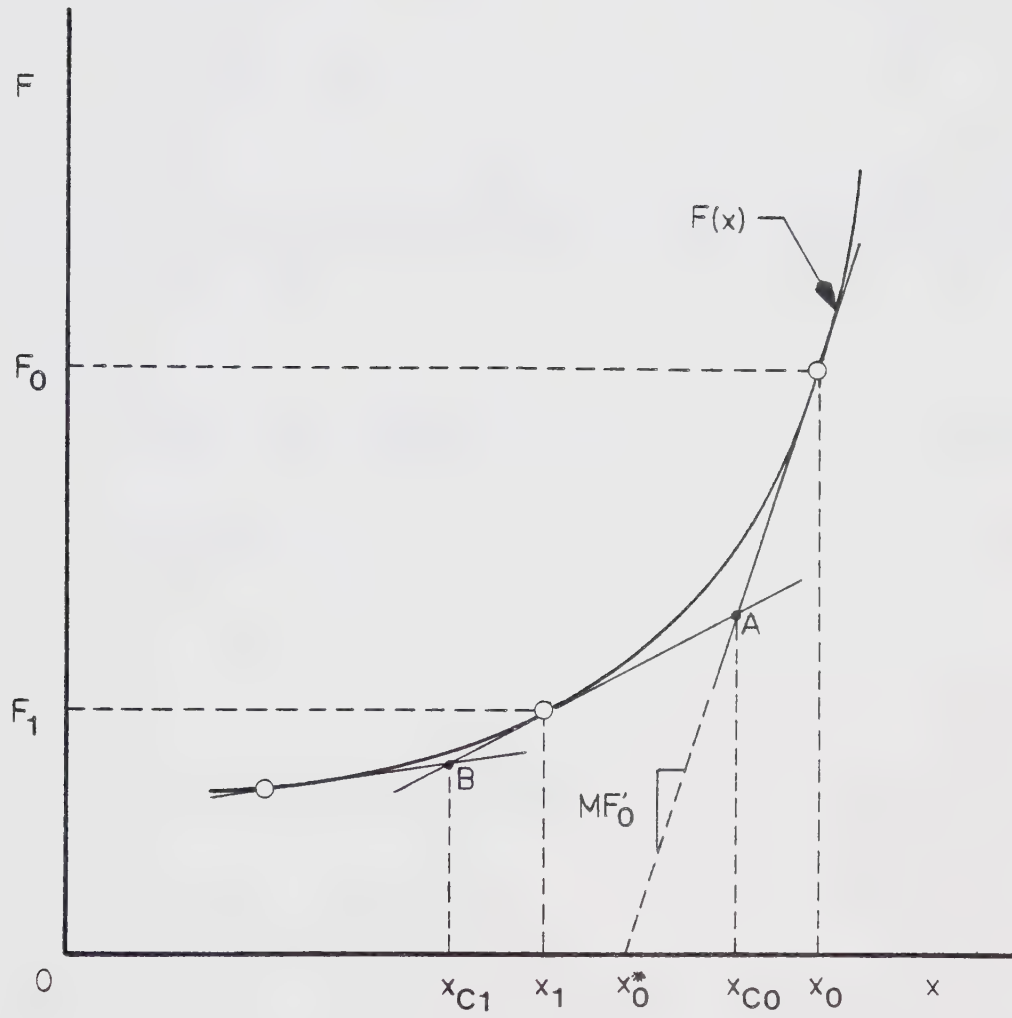


FIGURE 2.5 - TANGENT SEGMENT CONSTRUCTION

MF'_0 = Slope of tangent line at x_0 ,

F_0, F_1 = Same as for method of chords.

Again, as before for m equal sectors of length Δx (with m being an integer):

$$\begin{aligned}\Delta x &= \frac{x_0}{m} \\ &= x_0 - x_1.\end{aligned}$$

In this case, motion is from one crossing point to the next (that is, from A to B), and so, the equation for line AB is (for $x_{C1} \leq x \leq x_{C0}$ and $t_{C1} \geq t \geq t_{C0}$) [19]:

$$\frac{d^2}{dt^2}(x - x_0^*) + \frac{k_0}{M}(x - x_0^*) = 0, \quad (2.4.1)$$

$$\begin{aligned}\frac{k_0}{M} &= a + 3bx_0^2 \\ &= F'_0.\end{aligned}$$

From Figure 2.5 it can be seen that:

$$\begin{aligned}F'_0 &= \frac{F_0 - 0}{M(x_0 - x_0^*)} \\ &= \frac{ax_0 + bx_0^3}{x_0 - x_0^*} \\ x_0^* &= x_0 - \frac{ax_0 + bx_0^3}{a + 3bx_0^2}\end{aligned}$$

The solution to (2.4.1) is on the form:

$$x = x_0^* + A \cos p(t - t_{C0}) + B \cos p(t - t_{C0}) ,$$

where:

$$p = \sqrt{F_0'} .$$

The velocity is then [21]:

$$\dot{x} = -pA \sin p(t - t_{C0}) + pB \cos p(t - t_{C0}) .$$

Using the same method as before, but for $t = t_{C0}$, the solution for line AB is [21]:

$$\begin{aligned} x = x_0^* + (x_{C0} - x_0^*) \cos p(t - t_{C0}) \\ + \frac{\dot{x}_{C0}}{p} \sin p(t - t_{C0}), \end{aligned} \quad (2.4.2)$$

and the associated velocity [21]:

$$\begin{aligned} \dot{x} = -p(x_{C0} - x_0^*) \sin p(t - t_{C0}) \\ + \dot{x}_{C0} \cos p(t - t_{C0}) . \end{aligned} \quad (2.4.3)$$

Consider now what happens when the system reaches x_1 . In other words:

$$\begin{aligned} x_{C1} = x_0^* + (x_{C0} - x_0^*) \cos p(t_{C1} - t_{C0}) \\ + \frac{\dot{x}_{C0}}{p} \sin p(t_{C1} - t_{C0}) , \end{aligned}$$

$$\begin{aligned}\dot{x}_{C1} &= -p(x_{C0} - x_0^*) \sin p(t_{C1} - t_{C0}) \\ &+ \dot{x}_{C0} \cos p(t_{C1} - t_{C0}) .\end{aligned}$$

The first expression takes on the form of the trigonometric equation referred to in the previous section [22], with the coefficients in this case being:

$$\left. \begin{aligned}\alpha &= x_{C0} - x_0^* \\ &> 0 , \\ \beta &= \left| \frac{\dot{x}_{C0}}{p} \right| \\ &> 0 , \\ \gamma &= x_{C1} - x_0^* .\end{aligned} \right\} \quad (2.4.4)$$

Again, there are two possibilities for a solution with the same logic as before, such that, in this case, the result is [22]:

$$\begin{aligned}\theta &= \tan^{-1} \left(-\frac{\beta}{\alpha} \right) + \cos^{-1} \left(\frac{\gamma}{\sqrt{\alpha^2 + \beta^2}} \right) \\ &= p(t_{C1} - t_{C0}) , \\ t_1 &= t_0 + \frac{\theta}{p} .\end{aligned} \quad (2.4.5)$$

As before, the various segments have their individual values for t_1 , and so, the quarter-period is the sum of all of these.

However, there is a slight difference here, owing to the fact that at each point, a tangent is drawn. The endpoints will not have full tangent lines, as can be seen by inspection. For the initial

displacement, the equation of motion will be:

$$x = x_0^* + (x_{C0} + x_0^*) \cos pt .$$

For the end of the quarter period (or $x_1 = 0$), the result is:

$$\begin{aligned} -x_0^* &= (x_{C0} - x_0^*) \cos p(t_{C1} - t_{C0}) \\ &+ \frac{\dot{x}_{C0}}{p} \sin p(t_{C1} - t_{C0}) . \end{aligned}$$

Each x_{C0} is found as follows. From plane analytic geometry [24], for an x-y plot and two points (x_1, y_1) and (x_2, y_2) with the respective slopes being:

$$m_1 = \frac{y - y_1}{x - x_1} ,$$

$$m_2 = \frac{y - y_2}{x - x_2} ,$$

$$\begin{aligned} y &= m_1 x + (-m_1 x_1 + y_1) , \\ &= m_2 x + (-m_2 x_2 + y_2) . \end{aligned}$$

Extending this to the current situation:

$$\begin{aligned} F &= F_0' x_{C0} - F_0' x_0 + F_0 \\ &= F_1' x_{C0} - F_1' x_1 + F_1 , \\ (F_0' - F_1') x_{C0} &= F_0' x_0 - F_1' x_1 - (F_0 - F_1) , \\ x_{C0} &= \frac{F_0' x_0 - F_1' x_1 - (F_0 - F_1)}{F_0' - F_1'} , \end{aligned}$$

where:

$$F_1' = a + 3x_1^2 .$$

The two methods of solution had used different quantities for incrementation: time for the approximation to the exact solution and displacement for the piecewise linearization. For the former, time was the better of the two, as it is one variable of the argument. For the piecewise linearization, it was easier to use displacement, since one can readily find the values for x_0 , k_0 , and so on, since these are based upon a value for Δx . By using time, this does not become immediately apparent, and it would be difficult to determine the end of an interval. First the end of the segment would have to be guessed, and all the other variables used in the construction of a segment calculated on that basis. A variable, such as the current average segment velocity, is used as a means of determining the end of the interval by comparing it to the previous segment velocity, adjusting the end of the segment until there is an agreement, recalculating all the various variables concerned each time. This method is long and complicated and would increase the cost of the run. Also, the reference of Timoshenko, et. al. [18] already had a form of a solution that could readily be applied to this situation.

E. Results

A computer program was written using the three solutions, and the results will now be examined. This was run on the Amdahl 470V/8 at the University of Alberta [25], [26].

First consider the periods for each solution. It is important to consider this now as one is interested in the accuracy of the values obtained as compared with those from the approximation to the exact

TABLE 2.1 - UNDAMPED HARD SPRING EQUATION PERIODS
FOR $a = 1.0$, $b = 0.15$

Approx. to Exact Soln. = 5.958299670

<u>No. of Divisions</u>	<u>PIECEWISE LINEAR</u>	
	<u>CHORDS</u>	<u>TANGENTS</u>
5	5.951629401	5.961481246
10	5.956522358	5.959160535
15	5.957490299	5.958695097
20	5.957838384	5.958526637
25	5.958002069	5.958447100
30	5.958091944	5.958403305
35	5.958146564	5.958376633
40	5.958182232	5.958359186
45	5.958206807	5.958347149
50	5.958224456	5.958338494
55	5.958237560	5.958332062
60	5.958247556	5.958327151
65	5.958255356	5.958323316
70	5.958261559	5.958320265
75	5.958266573	5.958317797
80	5.958270684	5.958315772
85	5.958274097	5.958314090
90	5.958276961	5.958312678
95	5.958279389	5.958311481
100	5.958281464	5.958310457

(which uses the routine described in [4]). The first 100 terms of (2.2.4) were calculated for this. Initially, a low degree of nonlinearity was examined, with the results on Table 2.1. One can see, as was originally stated in the introduction, that as the number of segments gets large, the solution by chords and by tangents slowly approach the same value. Further evidence of this can be seen in Table 2.2, where the nonlinearity is more pronounced.

In theory, then, one could keep decreasing the size of the increments, and the periods would slowly converge to that of the exact solution to as many significant figures as is needed. The only limitation to this accuracy would be the execution time for the program, as each set of calculations will add to the run cost.

It should be noted that the period for the approximation to the exact solution was obtained as explained earlier in this chapter, while for the piecewise methods, the final value of t_1 was multiplied by 4.

An evaluation of the methods will be helped by examining the solution curves. The program that produced the results for Tables 2.1 and 2.2 also drew a plot each time it looped through, using the number of increments that was in effect as a basis.

The plots can be seen in Figures 2.6 through 2.9. A few comments should be made concerning the results. A good agreement to the approximation to the exact solution [4], [15], [17] (for $a = 1.0$ and $b = 0.15$) is obtained by both methods for 5 increments (Figure 2.6) and with 10 (Figure 2.7). Here one can hardly detect any difference between the piecewise method results and that of the approximation to (2.2.2) [4], [15], [17]. As one might suspect for $a = 1.0$ and $b = 2.0$, Figure 2.8 shows that there is a discernable deviation from the solution being

TABLE 2.2 - UNDAMPED HARD SPRING EQUATION PERIODS
FOR $a = 1.0$, $b = 2.0$

Approx. to Exact Soln. = 4.004308722

<u>No. of Divisions</u>	<u>PIECEWISE LINEAR</u>	
	<u>CHORDS</u>	<u>TANGENTS</u>
5	3.976017631	4.005261005
10	3.996812433	4.007978008
15	4.000907203	4.005978416
20	4.002378678	4.005261005
25	4.003061288	4.004923996
30	4.003438030	4.004739061
35	4.003666618	4.004626708
40	4.003815695	4.004553357
45	4.003918296	4.004502829
50	4.003991920	4.004466543
55	4.004046539	4.004439606
60	4.004088177	4.004419059
65	4.004120645	4.004403028
70	4.004146464	4.004390280
75	4.004167307	4.004379975
80	4.004184398	4.004371526
85	4.004198581	4.004364513
90	4.004210479	4.004358627
95	4.004220560	4.004353639
100	4.004229175	4.004349375

$a = 1.0000$ $b = 0.1500$
 NO. OF SEGMENTS (APPROX. TO EXACT
 SOLN.) = 100
 NO. OF CHORDS = 5
 NO. OF TANGENTS = 6
 $x(0) = 1.0000$ $\dot{x}(0) = 0.0000$
 — APPROX. TO EXACT SOLN.
 ○ CHORDS
 △ TANGENTS

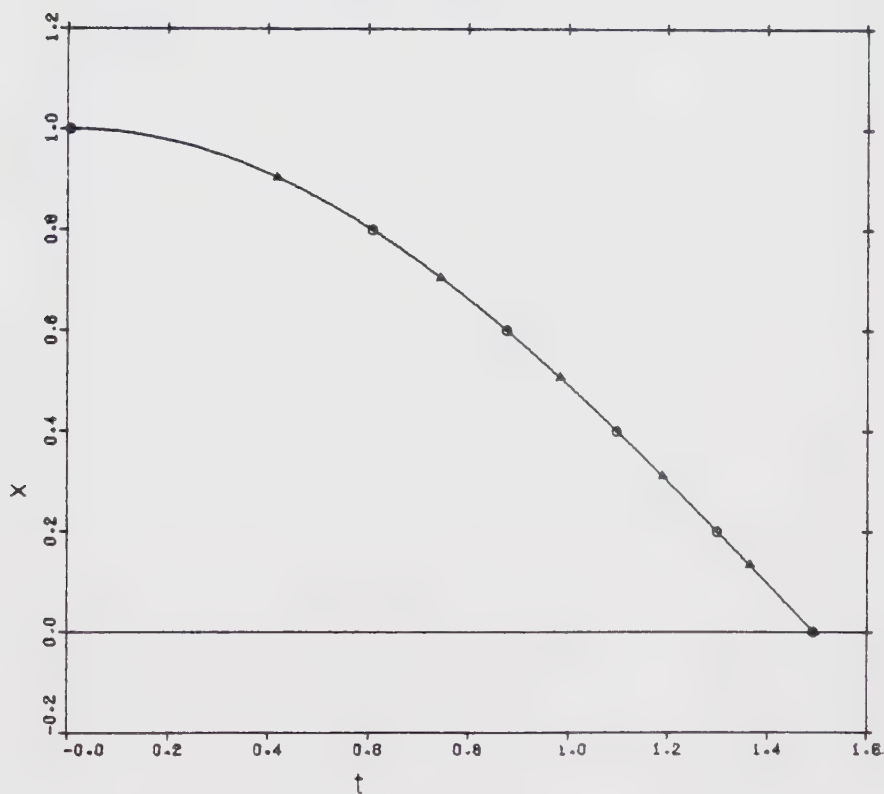


FIGURE 2.6 - SOLUTIONS TO HARD SPRING EQUATION
 (UNDAMPED) FOR $a = 1.0$, $b = 0.15$, $\Delta t = \tau_{\pi/4}/100.0$,
 $\Delta x = 0.20$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Approximation to exact solution uses [4], [15], [17].)

$a = 1.0000$ $b = 0.1500$
 NO. OF SEGMENTS (APPROX. TO EXACT
 SOLN.) = 100
 NO. OF CHORDS = 10
 NO. OF TANGENTS = 11
 $x(0) = 1.0000$ $\dot{x}(0) = 0.0000$
 — APPROX. TO EXACT SOLN.
 ○ CHORDS
 △ TANGENTS

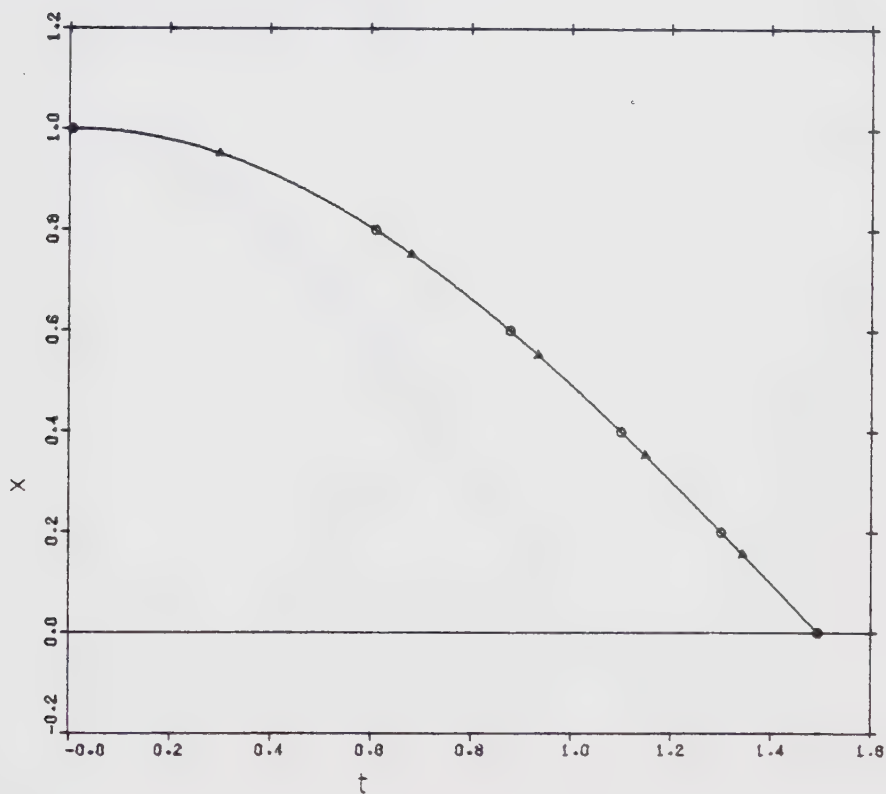


FIGURE 2.7 - SOLUTIONS TO HARD SPRING EQUATION
 (UNDAMPED) FOR $a = 1.0$, $b = 0.15$, $\Delta t = \tau_{\pi/4}/100.0$,
 $\Delta x = 0.10$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Approximation to exact solution uses [4], [15], [17].)

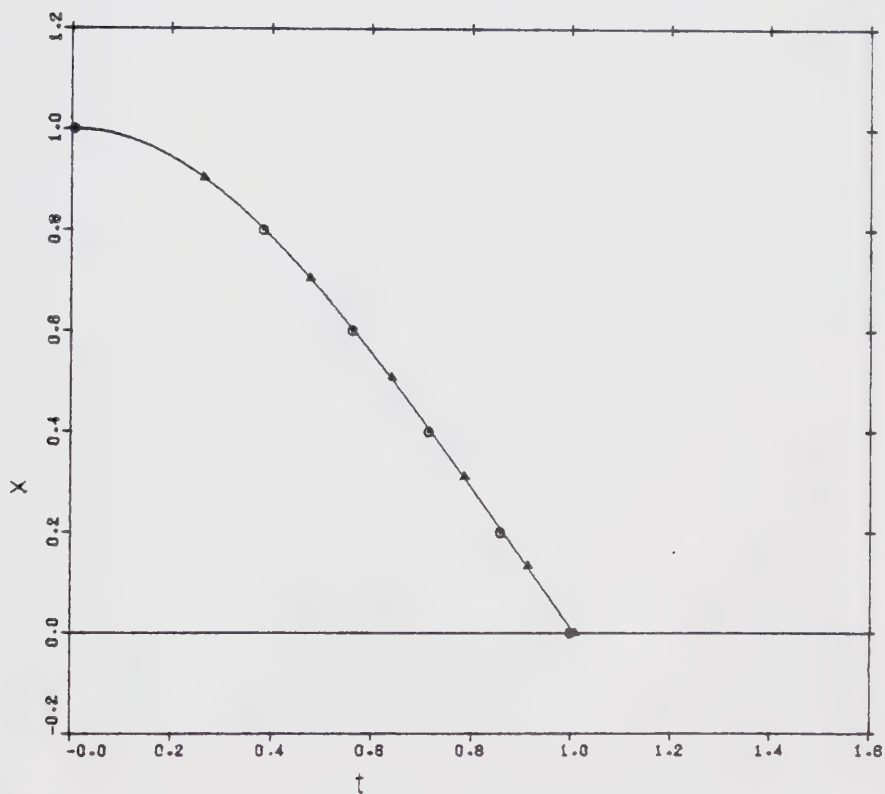
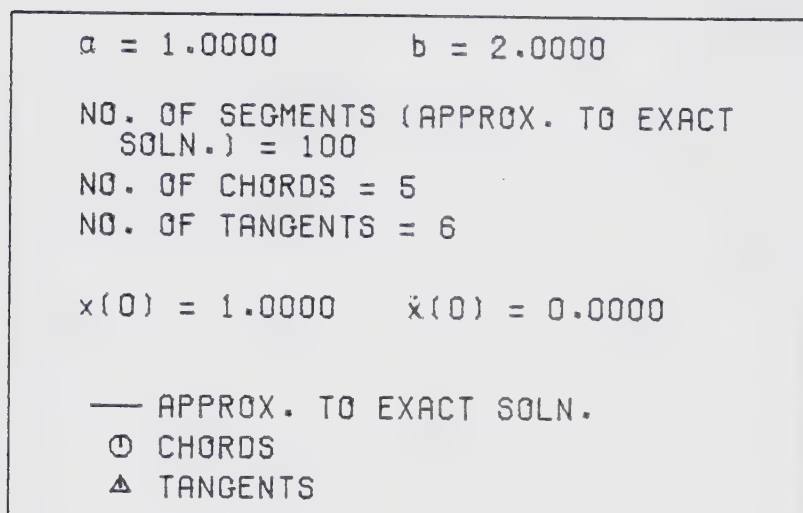


FIGURE 2.8 - SOLUTIONS TO HARD SPRING EQUATION
 (UNDAMPED) FOR $a = 1.0$, $b = 2.0$, $\Delta t = \tau_{\pi/4}/100.0$,
 $\Delta x = 0.20$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Approximation to exact solution uses [4], [15], [17].)

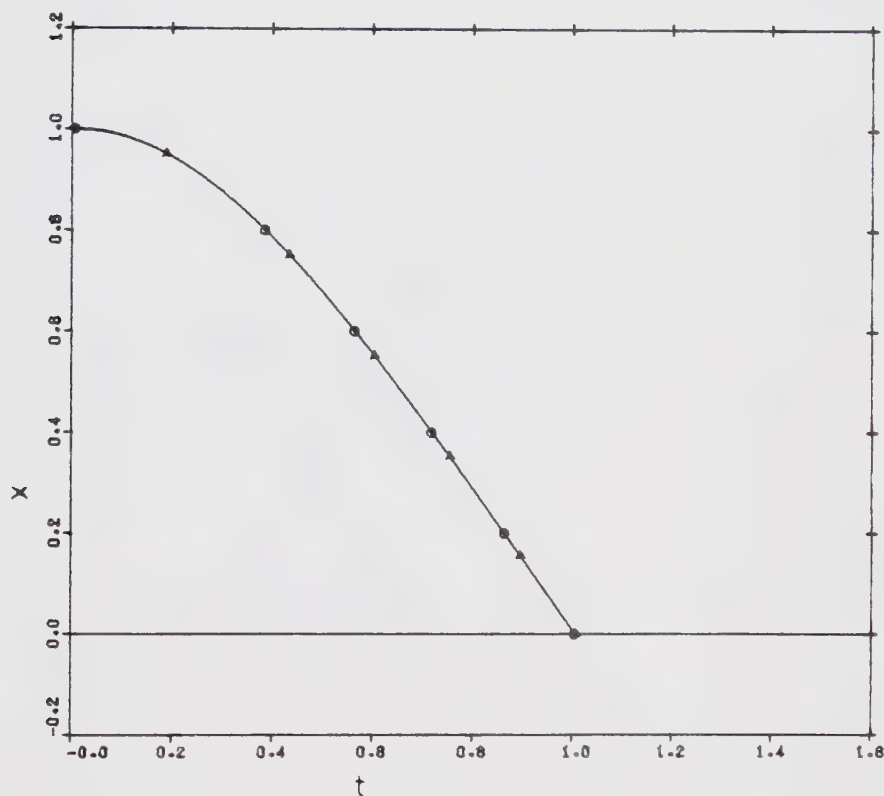
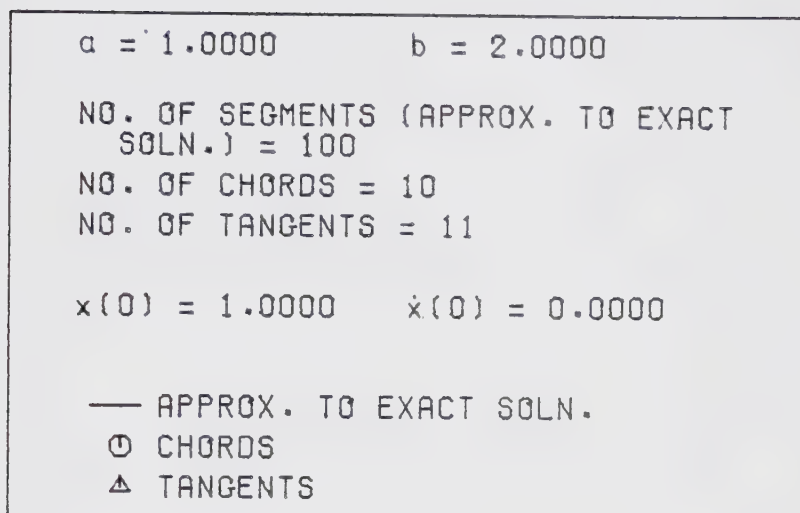


FIGURE 2.9 - SOLUTIONS TO HARD SPRING EQUATION
 (UNDAMPED) FOR $a = 1.0$, $b = 2.0$, $\Delta t = \tau_{\pi/4}/100.0$,
 $\Delta x = 0.10$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Approximation to exact solution uses [4], [15], [17].)

compared, since the degree of nonlinearity is considerably higher than in the first case. This does not appear in Figure 2.9, where 10 increments were used.

A question that should be raised is concerning the actual execution times involved, since, when choosing a method of solving these sorts of problems, one must strike a balance between required accuracy to the solution being compared and the cost of the run. This was done by taking the original program and modifying it by stripping it of all non-essential operations with only the actual calculations remaining, and then utilizing a timing subroutine available through the library of the computer system, and calling it for each solution [27]. The results are seen on Figures 2.10 and 2.11. It should be noted here that these results were obtained with an older and somewhat slower version of the the program, but they should serve to illustrate the relative execution times. It should also be noted that these results were obtained with an Amdahl 470V/7 [25]. This difference in the computers was due to an upgrading of the original 470V/7 model to a 470V/8 done while the research for this thesis was being carried out [26]. The author reran the timing program and found that the results were about 10 percent faster with the V/8 than the V/7. Also, with some further minor modifications to the program itself, the author estimates about another 10 percent can be taken off the original times.

By inspection, one can see that the actual distance travelled by the system using the method of chords would be somewhat less than that using tangents. Already this implies that the period would be greater using the latter method than the former when comparing it to that obtained by using the results based on the exact solution. That would

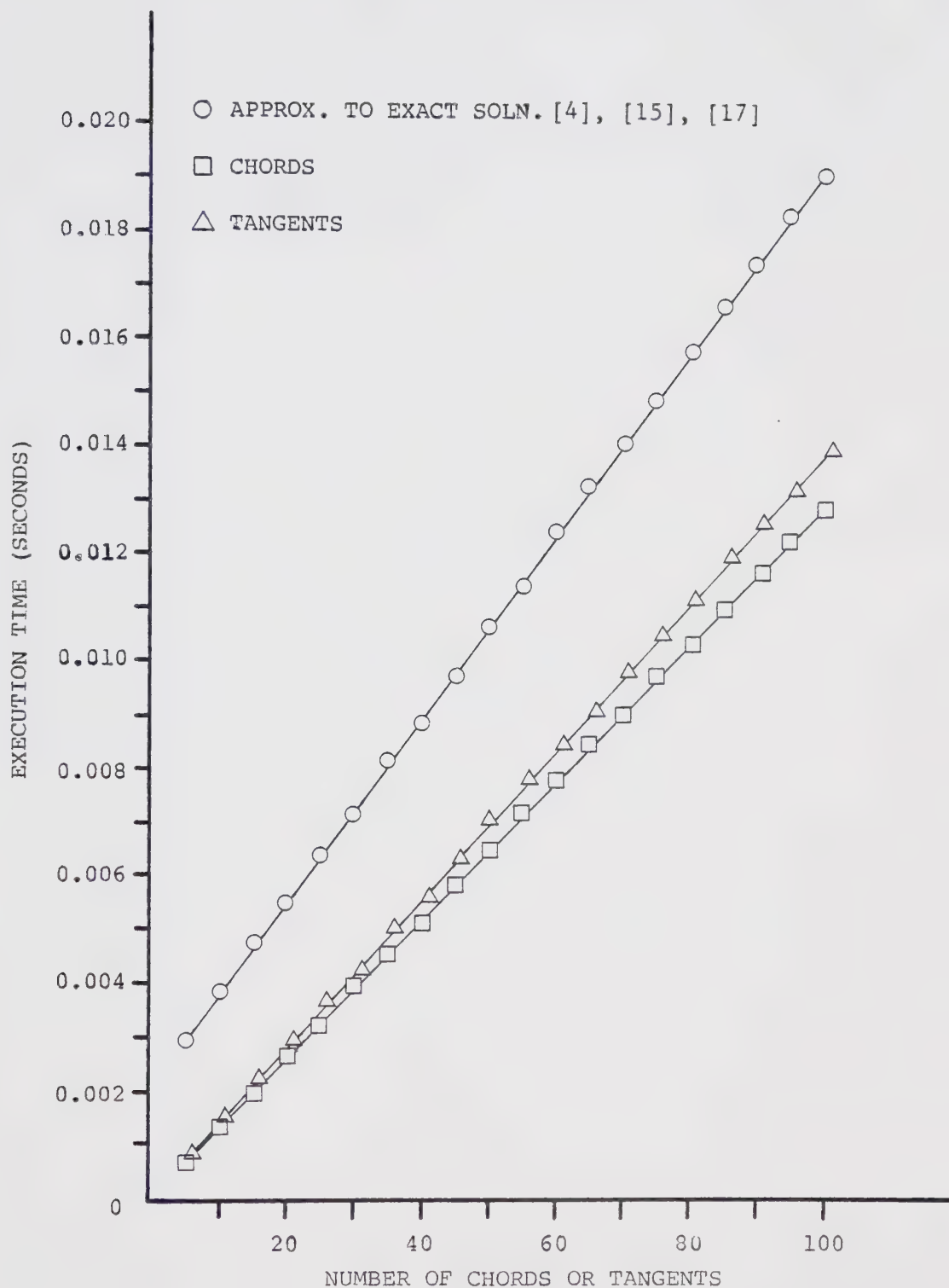


FIGURE 2.10 - EXECUTION TIMES FOR SOLUTIONS TO HARD SPRING EQUATION (UNDAMPED) FOR $a = 1.0$, $b = 0.15$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$ [27]

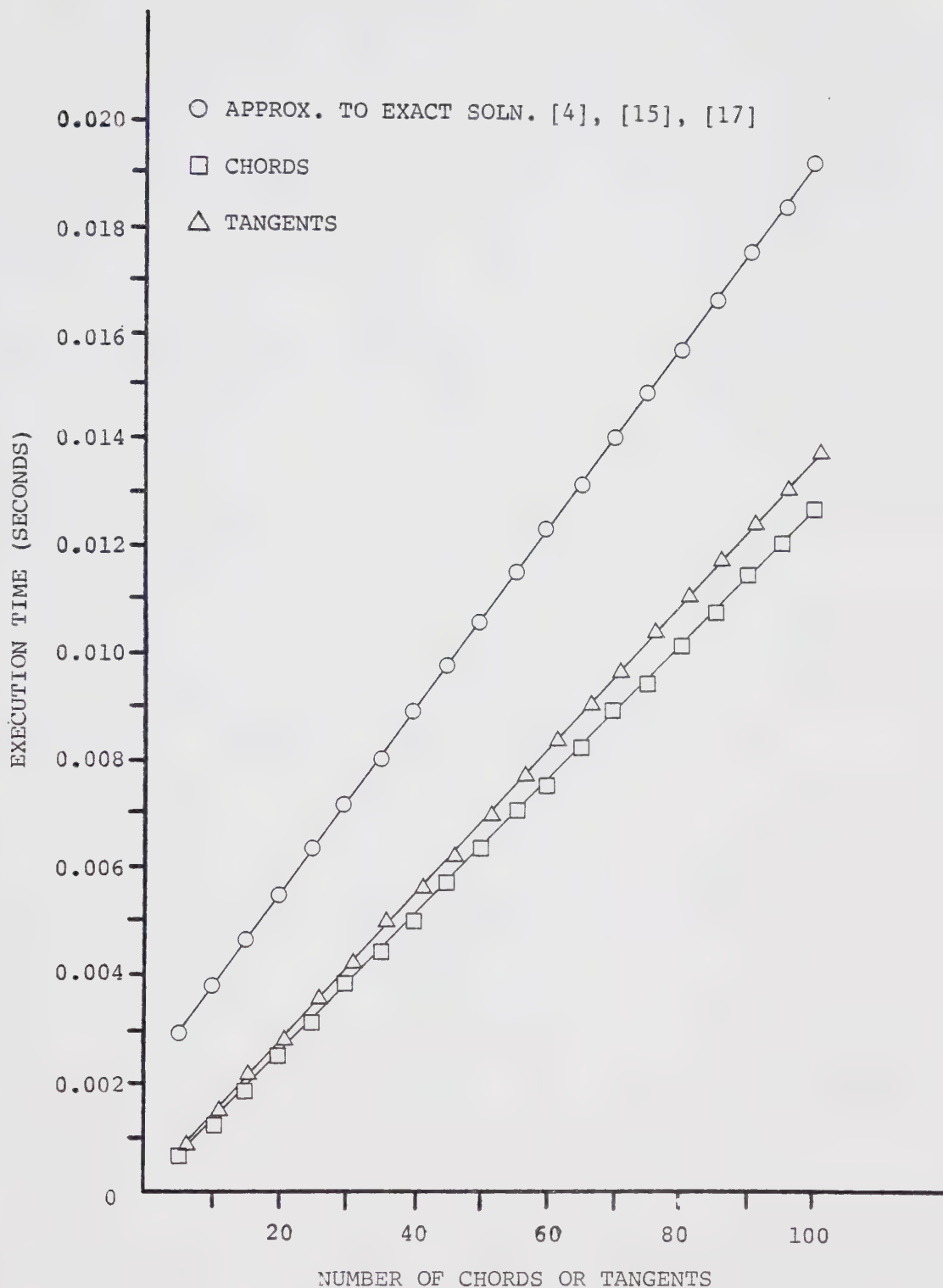


FIGURE 2.11 - EXECUTION TIMES FOR SOLUTIONS TO HARD SPRING EQUATION (UNDAMPED) FOR $a = 1.0$, $b = 2.0$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$ [27]

explain why, when b was 2.0, for example, one obtained a difference in the displacement on the higher side using tangents than chords.

Another question is why the timing results for the approximation to the exact solution [4], [15], [17] should be considerably higher than that for the piecewise methods. A word of explanation concerning the program will clarify this point. The execution times were obtained by calculating the first 50 terms of (2.2.4)[15] and then generating the solution results, prior to increasing the number of segments and looping through the calculations once again. This was done to determine the total time required to obtain a solution for a given Δx , thus allowing comparison with the other methods. This required that the subroutine that calculated (2.2.4)[15] had to be called each time.

Since more calculations are necessary for each segment for the method of tangents, it would be expected that the execution times would be greater for this solution than for the method of chords.

In any case, the results for the piecewise methods are impressive when one considers that for this problem, it would appear that an accurate solution can be easily obtained by using either method, with a relatively small number of segments, and that it is faster than the solution that was chosen to be the standard one [4], [15], [17]. One can conclude that for this problem, the piecewise method would be the recommended one to use, particularly the method by chords.

Because the method of chords was easier to manipulate and converged faster to the approximation to (2.2.2)[4], [15], [17], it was chosen to be the method of piecewise linearization for the remainder of the thesis and will subsequently be referred to as such.

III. HARD SPRING EQUATION (LINEAR DAMPING)

A. Preliminary Comments

Most physical systems involve dissipation, and an example of this is the system described in the previous chapter, but with the inclusion of a simple form of damping in which it is linearly related to the velocity:

$$\ddot{x} + \frac{c}{M} \dot{x} + ax + bx^3 = 0 , \quad (3.1.1)$$

where:

c = Damping constant,

M = System mass,

a, b = Restoring force factors, as used in
Chapter II.

The system is shown in Figure 3.1 [28].

Since there is no exact solution and approximate methods such as those mentioned in the first chapter are beyond the scope of this thesis, a Runge-Kutta solution [5] has been chosen to be the standard by which to judge the accuracy of the piecewise method.

For convenience, it was decided to restrict the cases examined to underdamped conditions.

The values for a and b are the same as in the previous chapter, while $\frac{c}{M}$ will have the values of 0.0, 0.1, and 0.25 which should give a good representation of an underdamped case.

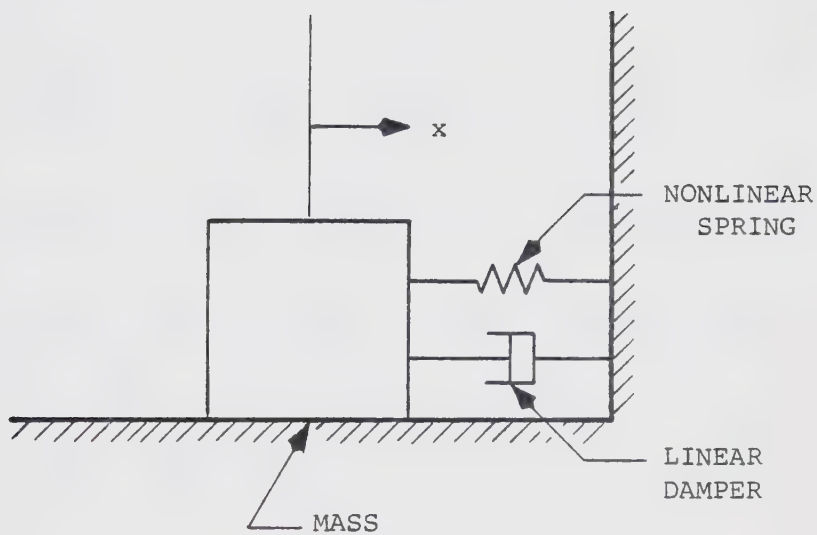


FIGURE 3.1 - MASS-SPRING SYSTEM (LINEAR DAMPING)[28]

B. Derivation of Piecewise Linear Solution

Since the piecewise linearization method by chords had been chosen to be the means of solving the equations for the remainder of the thesis, and much of the background derivation is exactly the same as before, only the results of this plus the derivation of new material will be shown in this and subsequent chapters.

Recalling the development of the method by chords and using the same initial conditions and notation as in the previous chapter, the following equation results for a linear segment [19], [29]:

$$\frac{d^2}{dt^2} (x - x_0^*) + \frac{c}{M} \dot{x} + \frac{k_0}{M} (x - x_0^*) = 0,$$

k_0 = Slope of chord line, as in Chapter II.

where x , x_0 , and \dot{x} are the same as before and are found in the manner previously described in Chapter II, Section C. The solution to this system is now the familiar [29]:

$$x = x_0^* + e^{-n(t - t_0)} \left[A \cos p^*(t - t_0) + B \sin p^*(t - t_0) \right], \quad (3.2.1)$$

where:

$$p^* = \sqrt{\frac{k_0}{M} - n^2},$$

t_0 = Time taken for system to reach current x_0 ,

$$n = \frac{c}{2M},$$

$$A = x_0 - x_0^*,$$

$$B = \frac{\dot{x}_0 + n(x_0 - x_0^*)}{p^*} .$$

Differentiating (3.2.1) yields the velocity [29]:

$$\begin{aligned} \dot{x} = e^{-n(t - t_0)} & \left[(p^*B - nA) \cos p^*(t - t_0) \right. \\ & \left. + (-nB - p^*A) \sin p^*(t - t_0) \right] , \end{aligned} \quad (3.2.2)$$

with the various coefficients as before.

Solving for $t_1 - t_0$ requires the use of a Newton-Raphson method, since there does not appear to be a simpler way of finding this quantity such as that used in the previous chapter. This would be [30]:

$$(t_1 - t_0)_{i+1} = (t_1 - t_0)_i - \frac{x[(t_1 - t_0)_i]}{\dot{x}[(t_1 - t_0)_i]} .$$

However, as the displacement approaches the first trough, the velocity approaches zero, and consequently the method as defined would break down. But, since this can be set up in a computer program, a check can be run such that when the denominator becomes very small (corresponding to a small speed) the Newton-Raphson relationship can be redefined as [30]:

$$(t_1 - t_0)_{i+1} = (t_1 - t_0)_i - \frac{\dot{x}[(t_1 - t_0)_i]}{x[(t_1 - t_0)_i]} ,$$

$$\ddot{x}[(t_1 - t_0)_i] = e^{-n[(t_1 - t_0)_i]} \cdot \{ [(-p^2 + n^2)A - 2npB] \cos p[(t_1 - t_0)_i] + [2npA + (-p^2 + n^2)B] \sin p[(t_1 - t_0)_i] \} ,$$

with the coefficients the same as used for (3.2.1) and (3.2.2).

Also, there is the likelihood that the first trough to be encountered bottoms out in the middle of a segment, which poses another problem as to how to determine exactly where this occurs. First $t_1 - t_0$ is found as before (with the second Newton-Raphson relationship being in effect as well as \dot{x} will approach zero). The appropriate value for the displacement is found, and this is compared to what the particular x_1 would have been. If the difference between the two values is small, then the iterations stops. If not, this x becomes the new x_1 , and new end conditions are found with the iteration continuing using $t_1 - t_0$.

The half-period, then, would be the final value of t_1 .

C. Runge-Kutta Solution

The Runge-Kutta numerical method used throughout this thesis is the routine DVERK from IMSL [5]. Since it was in a system library, it was easily accessed by the program written to solve this situation. The basis of the method is an approach using approximations described in this reference.

D. Results

A computer program was written that included the piecewise linear solution, complete with the necessary subroutines for calculating the

FIGURE 3.2 - SOLUTIONS TO HARD SPRING EQUATION
(LINEAR DAMPING) FOR $a = 1.0$, $b = 0.15$, $\frac{c}{M} = 0.0$,
 $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5].)

$a = 1.0000$ $b = 0.1500$
 $c/M = 0.0000$

Δx (PIECEWISE
 LINEARIZATION) = 0.1000
 Δt (RUNGE-KUTTA) = 0.0500

$x(0) = 1.0000$ $\dot{x}(0) = 0.0000$

— RUNGE-KUTTA
 ○ PIECEWISE LINEARIZATION

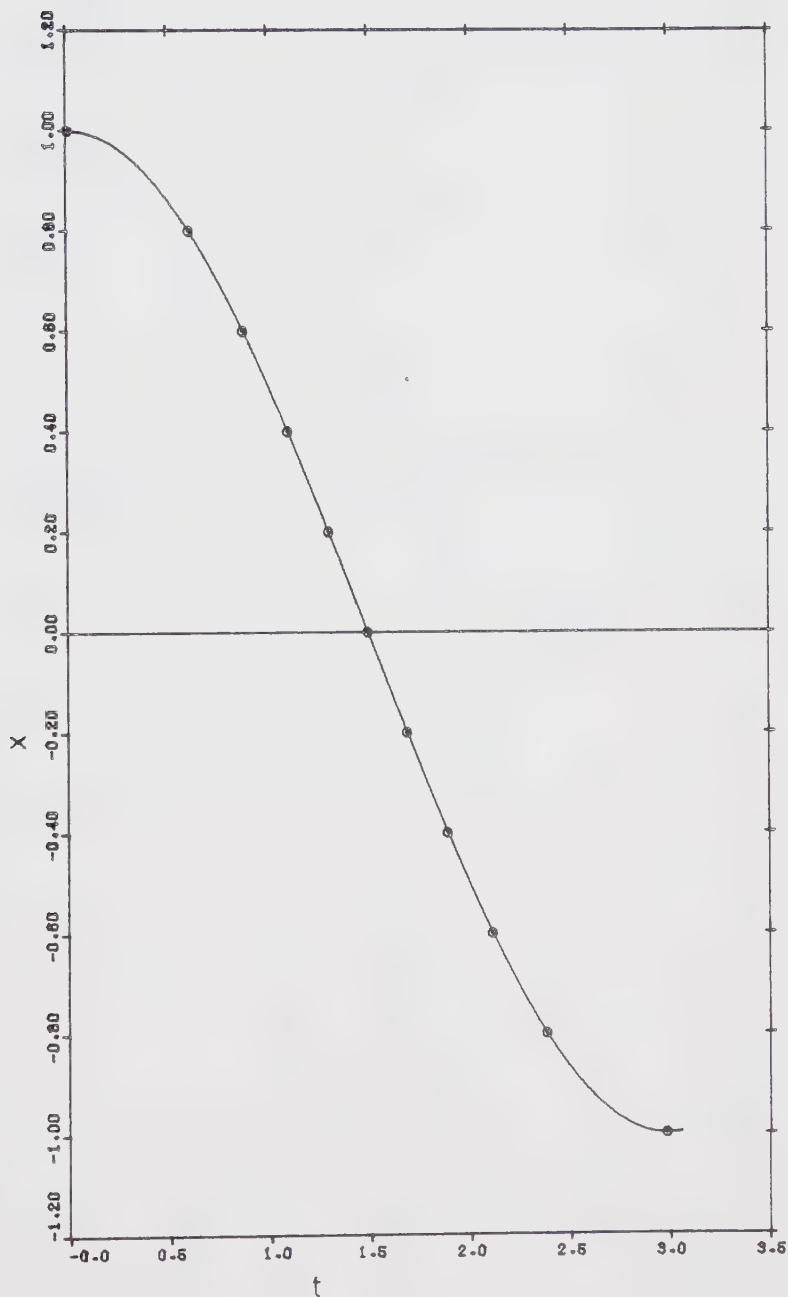


FIGURE 3.3 - SOLUTIONS TO HARD SPRING EQUATION
(LINEAR DAMPING) FOR $a = 1.0$, $b = 2.0$, $\frac{c}{M} = 0.0$,
 $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5].)

$$a = 1.0000 \quad b = 2.0000$$

$$c/M = 0.0000$$

$$\Delta x \text{ (PIECEWISE LINEARIZATION)} = 0.1000$$

$$\Delta t \text{ (RUNGE-KUTTA)} = 0.0500$$

$$x(0) = 1.0000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

○ PIECEWISE LINEARIZATION

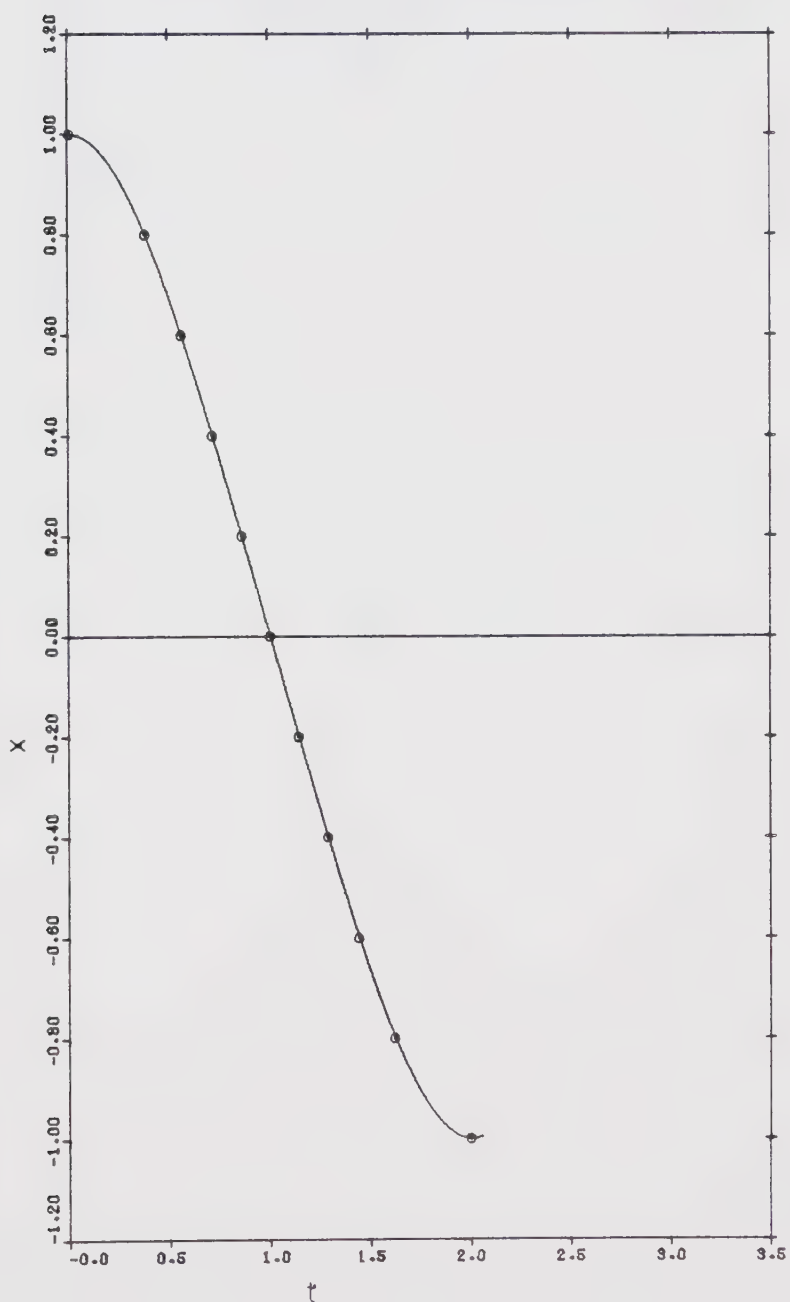


FIGURE 3.4 - SOLUTIONS TO HARD SPRING EQUATION
(LINEAR DAMPING) FOR $a = 1.0$, $b = 0.15$, $\frac{c}{M} = 0.10$,
 $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5].)

$$a = 1.0000 \quad b = 0.1500$$

$$c/M = 0.1000$$

$$\Delta x \text{ (PIECEWISE LINEARIZATION)} = 0.1000$$

$$\Delta t \text{ (RUNGE-KUTTA)} = 0.0500$$

$$x(0) = 1.0000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

⊙ PIECEWISE LINEARIZATION

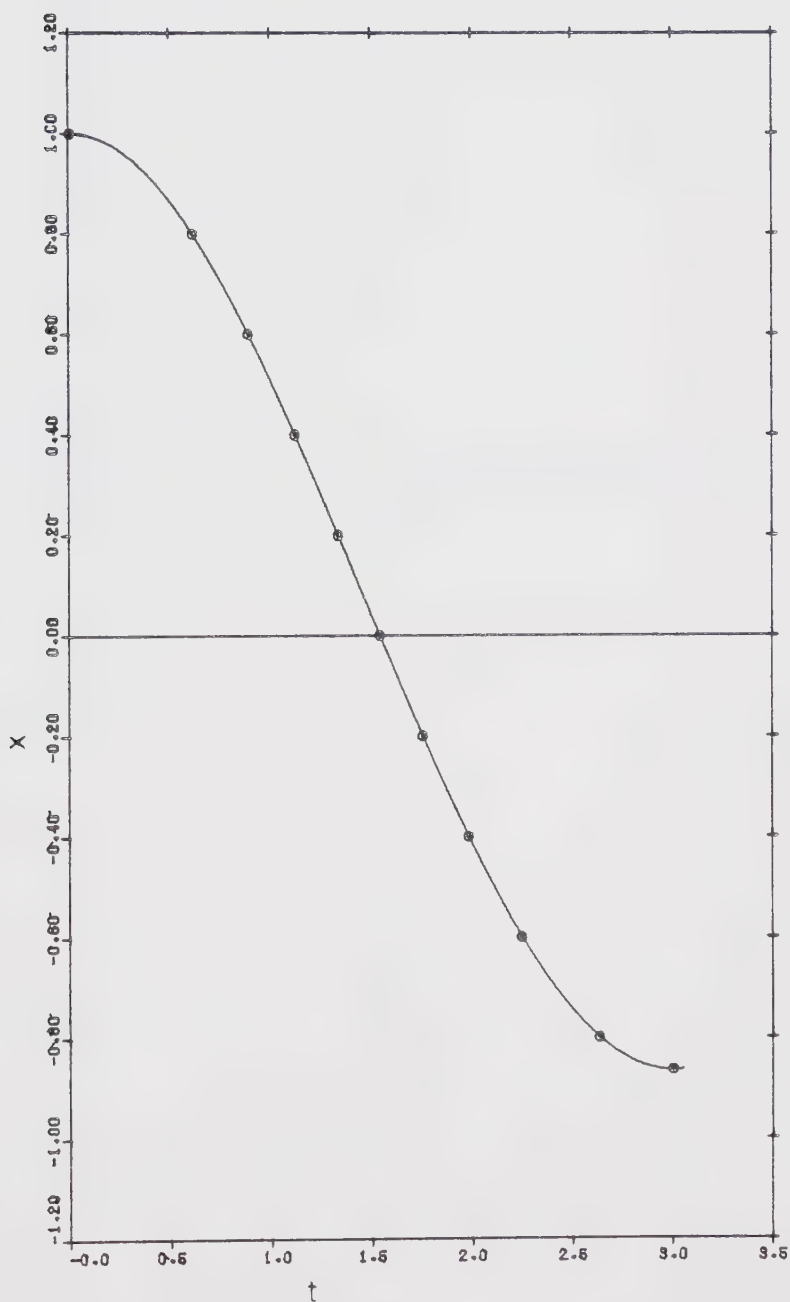


FIGURE 3.5 - SOLUTIONS TO HARD SPRING EQUATION
(LINEAR DAMPING) FOR $a = 1.0$, $b = 2.0$, $\frac{c}{M} = 0.10$,
 $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5].)

$a = 1.0000$ $b = 2.0000$
 $c/M = 0.1000$

Δx (PIECEWISE
 LINEARIZATION) = 0.1000

Δt (RUNGE-KUTTA) = 0.0500

$x(0) = 1.0000$ $\dot{x}(0) = 0.0000$

— RUNGE-KUTTA

○ PIECEWISE LINEARIZATION

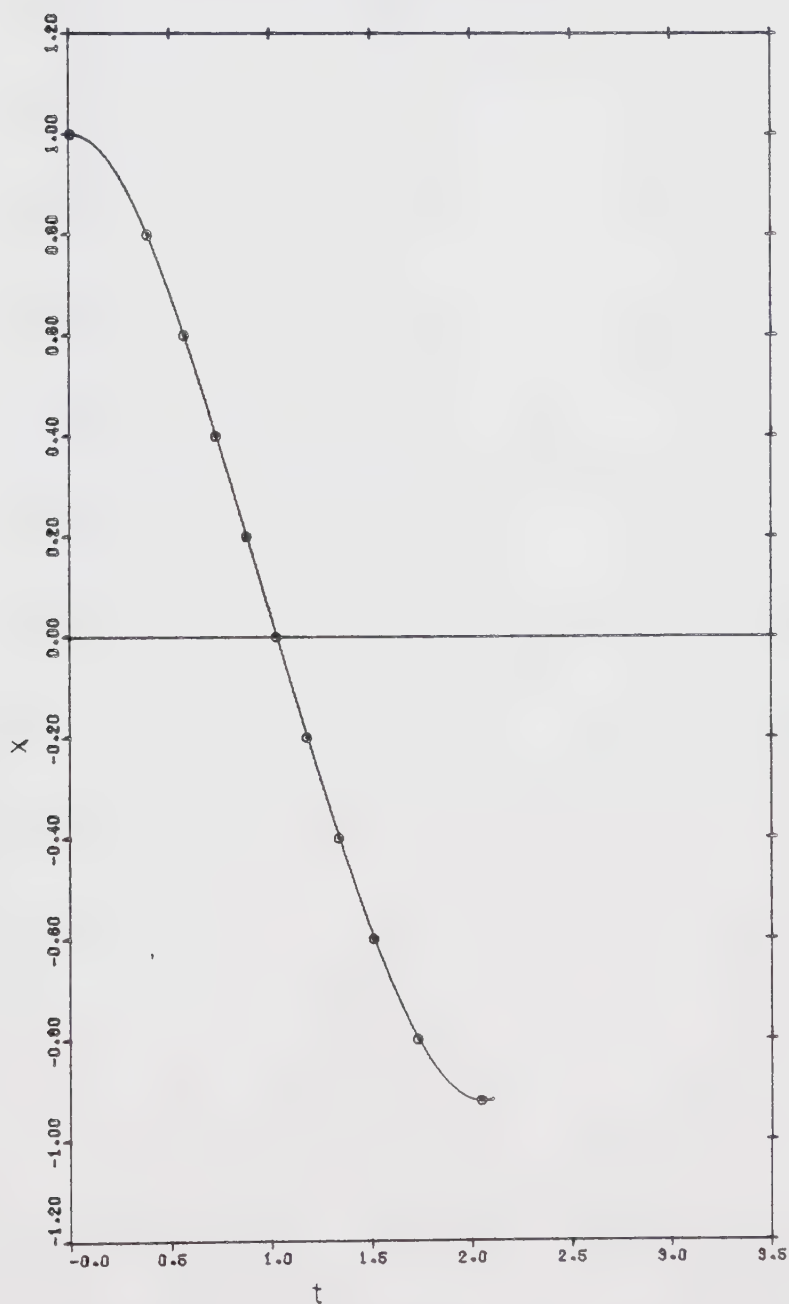


FIGURE 3.6 - SOLUTIONS TO HARD SPRING EQUATION
(LINEAR DAMPING) FOR $a = 1.0$, $b = 0.15$, $\frac{c}{M} = 0.25$,
 $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5].)

$$a = 1.0000 \quad b = 0.1500$$

$$c/M = 0.2500$$

$$\Delta x \text{ (PIECEWISE LINEARIZATION)} = 0.1000$$

$$\Delta t \text{ (RUNGE-KUTTA)} = 0.0500$$

$$x(0) = 1.0000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

○ PIECEWISE LINEARIZATION

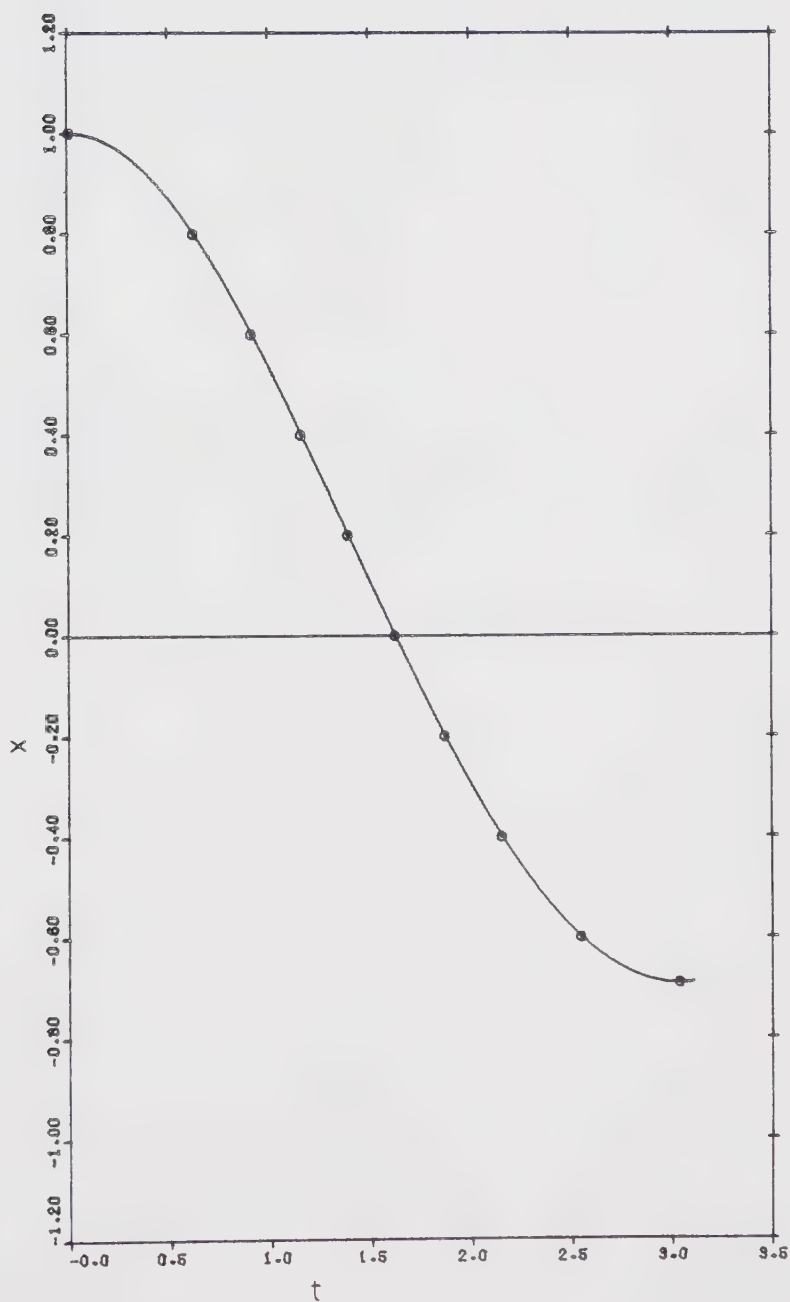


FIGURE 3.7 - SOLUTIONS TO HARD SPRING EQUATION
(LINEAR DAMPING) FOR $a = 1.0$, $b = 2.0$, $\frac{c}{M} = 0.25$,
 $\Delta x = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5].)

$$a = 1.0000 \quad b = 2.0000$$

$$c/M = 0.2500$$

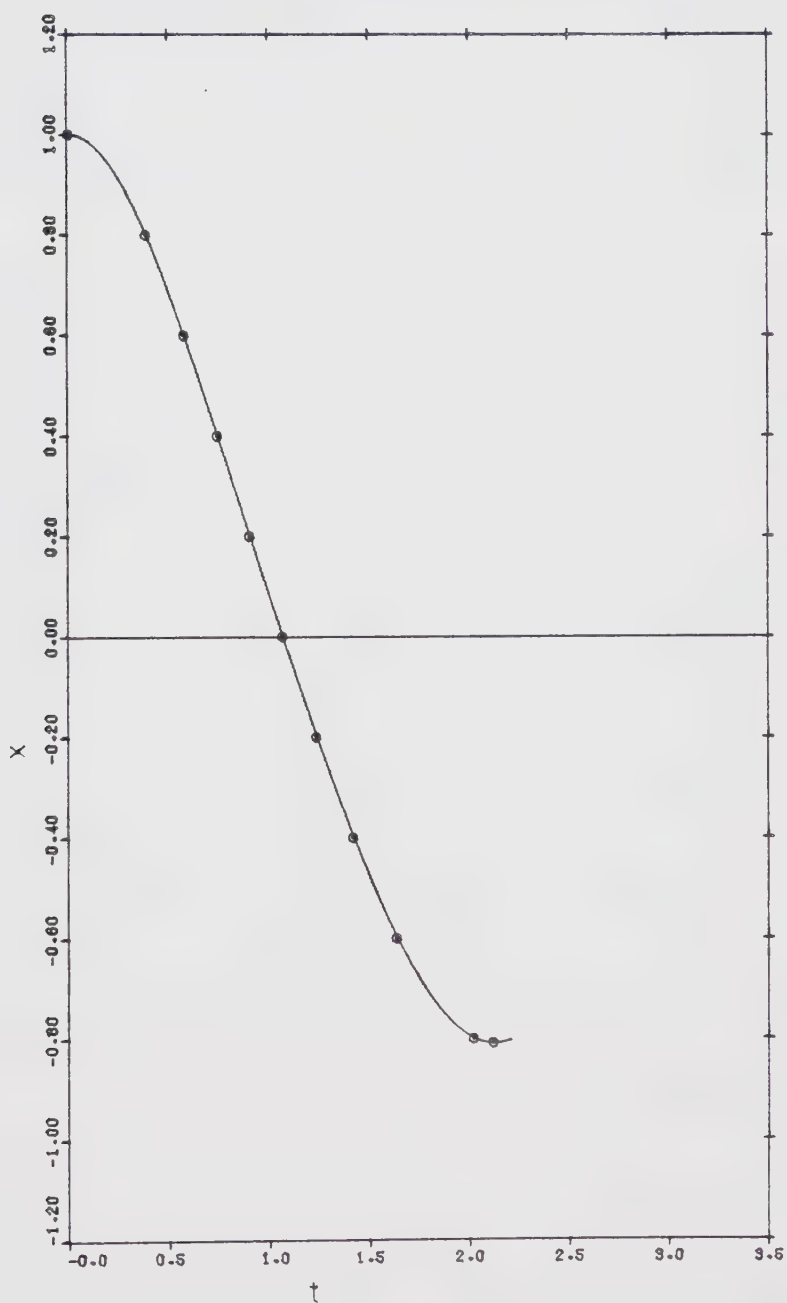
$$\Delta x \text{ (PIECEWISE LINEARIZATION)} = 0.1000$$

$$\Delta t \text{ (RUNGE-KUTTA)} = 0.0500$$

$$x(0) = 1.0000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

⊙ PIECEWISE LINEARIZATION



various parameters like x_0^* , as well as the input parameters and call to DVERK [5]. The results were obtained by running the program on the Amdahl 480V/8 [26] described earlier.

As a first trial, to see if the program functioned, it was decided to check if the solution would work for the simple case of $\frac{C}{M}$ being zero for the sets of values for a and b as described earlier in the chapter, and X_0 being taken to be 1.0. The results of this are seen in Figures 3.2 and 3.3, using a value of Δx of 0.1, and a time increment for the Runge-Kutta solution [5] of 0.05, the sizes being chosen arbitrarily, but still sufficiently large to allow any problems concerning stability arise.

From Figures 3.2 and 3.3, it is obvious that the program works for the undamped case, as the correspondence between the two solutions is good, and there appears to be little problem in the piecewise linear method bottoming out at the first trough.

Now for solving the equation using non-zero values for $\frac{C}{M}$. The results for this can be seen in Figures 3.4 - 3.7. For each plot there seems to be a good correspondence between the results obtained for the piecewise linear solution and the Runge-Kutta procedure [5], which implies two things. One is that the method of solution using the piecewise linearization by chords gives a good agreement with the Runge-Kutta approximation [5] of this equation. The other is that for even a somewhat large increment of displacement, the piecewise linear method gives a good result, and that it is not necessary to go to a smaller value to get an accurate solution. This has an advantage as far as the economics of the runs are concerned.

As before, the program was stripped down to the essential

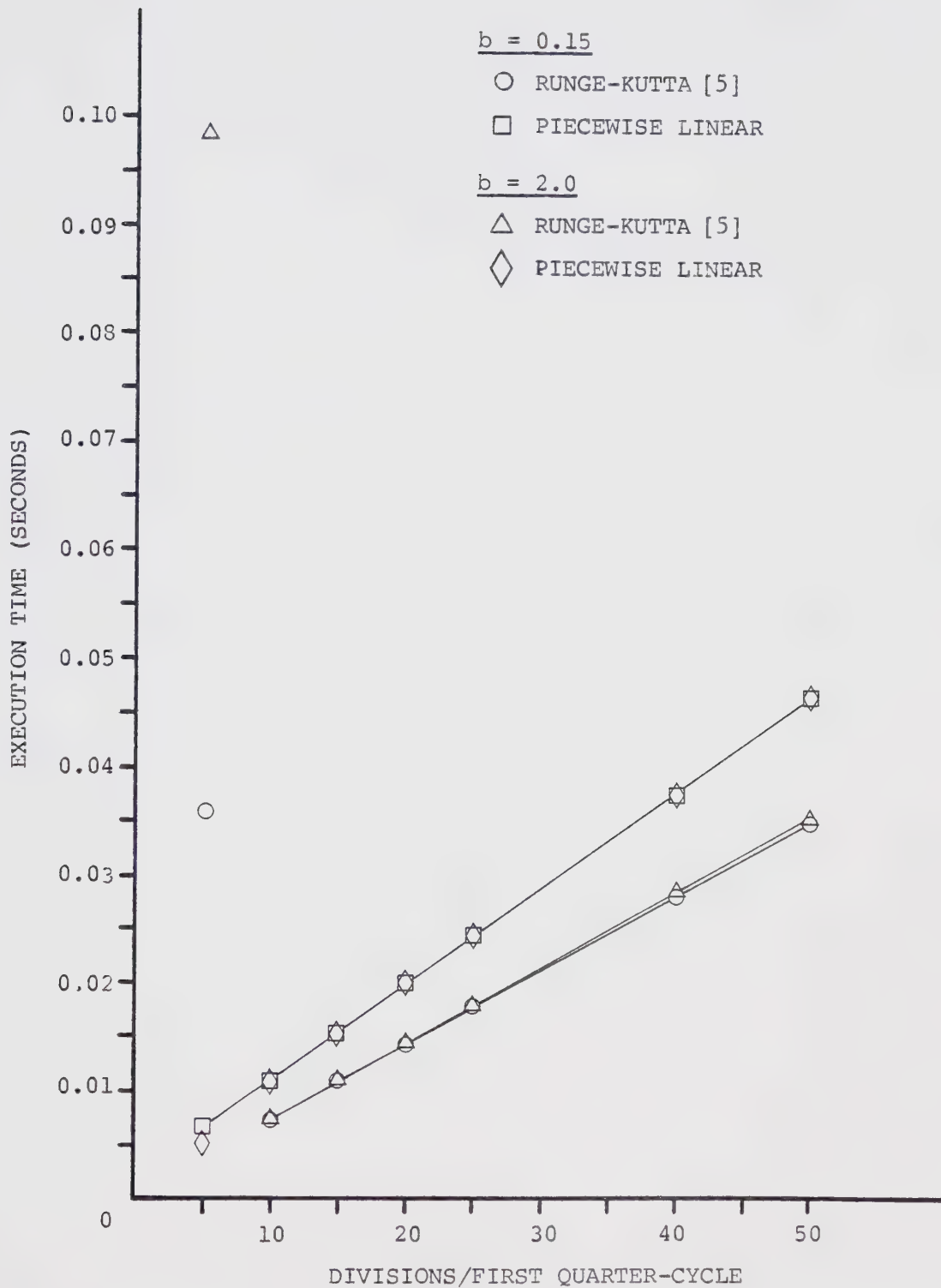


FIGURE 3.8 - EXECUTION TIMES FOR SOLUTIONS TO HARD SPRING EQUATION (LINEAR DAMPING) FOR
 $a = 1.0, \frac{c}{M} = 0.0, x(0) = 1.0, \dot{x}(0) = 0.0$ [27]

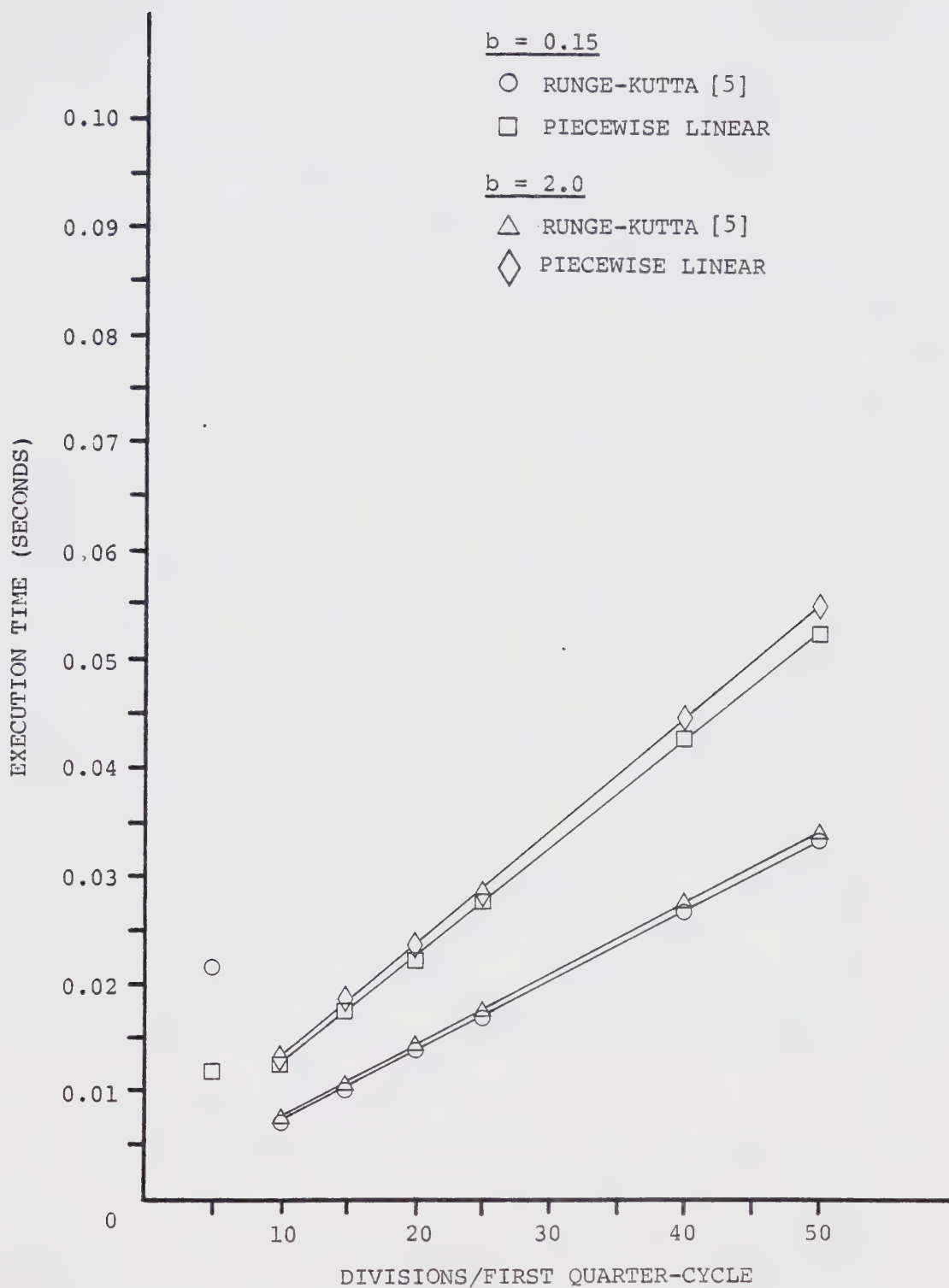


FIGURE 3.9 - EXECUTION TIMES FOR SOLUTIONS TO HARD SPRING EQUATION (LINEAR DAMPING) FOR
 $a = 1.0$, $\frac{c}{M} = 0.10$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$ [27]

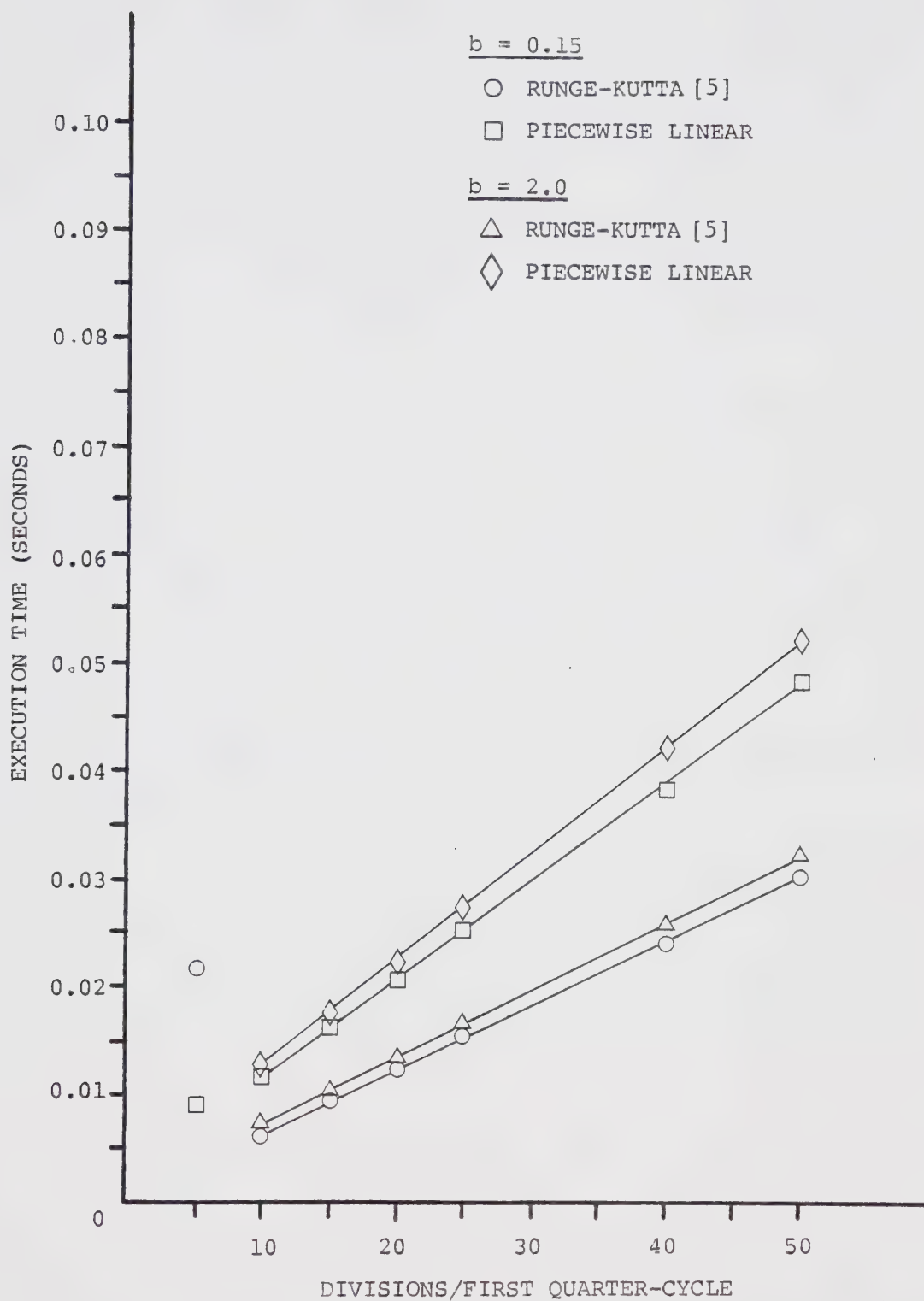


FIGURE 3.10 - EXECUTION TIMES FOR SOLUTIONS TO HARD SPRING EQUATION (LINEAR DAMPING) FOR
 $a = 1.0$, $\frac{c}{M} = 0.25$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$ [27]

calculations and the execution times for each method of solutions obtained [27]. Again, it should be noted that these values were acquired with an older version of the plot program that was slower and that this was done on the V/7 version of the Amdahl 470 [25], [26]. The results can be seen in Figures 3.8 - 3.10.

Before continuing, an explanation is necessary concerning the timing results. As mentioned earlier, the piecewise method used an interval of displacement while the Runge-Kutta method [5] made use of time. For the purposes of determining correspondence between the two solutions, this would be sufficient, but this would not be valid for comparing the execution times, since Δx and Δt would scarcely provide any means for judgement between the two methods. However, the number of divisions (or overall iterations) for the first quarter-cycle would possibly be a benchmark. To an extent, this was done for the piecewise method, since the system initially had a known displacement, and Δx was chosen from that.

For the Runge-Kutta method [5], a similar approach can be used based upon the first quarter-period as calculated by the piecewise method. This timespan can be divided into the same number of intervals as was used by the first solution, and the execution time is obtained with the new value for Δt .

The timing program kept track of the number of points calculated by the piecewise solution (including the first trough, which generally falls inside an interval) based on Δx while the Runge-Kutta routine [5] calculates this same number of points, but using the new Δt as was just described. This should give a reasonable correspondence to the piecewise solution except that the Runge-Kutta method [5] will likely go past

the first trough as before.

One might expect a linear relationship between the execution time and the number of divisions of the first-quarter cycle. Since the Runge-Kutta solution [5] is from a packaged program, presumably it should be as efficient as possible, which would be a reasonable explanation as to why the execution times for it are lower as compared to the piecewise linear method. However, what is of note is what happens at 5 divisions.

For some of the runs, no values were obtained, as the execution time limit that had been set prior to the run had been exceeded. Attempts to restart the program by adding additional seconds to bring the total execution time limit to at least 15 seconds failed to give results [31]. Some clue may be available from those runs that used what may appear to be excessive run time values.

For runs such as for $a = 1.0$, $b = 0.15$, and $\frac{C}{M} = 0.25$, the points are far off the line. This may indicate that there could be a lower limit for which either solution would work (or be economical), as from what was obtained, one can see that the Runge-Kutta [5] took longer to finish at this point than the piecewise linear method, which was not expected.

IV. HARD SPRING EQUATION (NONLINEAR DAMPING)

A. Preliminary Comments

This situation is essentially the same as that examined in the previous chapter, with the exception that the damping is no longer linear, but is related to the square of the velocity. The solution for the linear segments is the same, but the means by which it is obtained are slightly different. The equation considered is:

$$\ddot{x} + \frac{q}{M} |\dot{x}|^2 \operatorname{sgn}(\dot{x}) + ax + bx^3 = 0, \quad (4.1.1)$$

where:

q = Nonlinear damping constant,

M, a, b = as used in Chapter III.

This equation describes the system shown in Figure 4.1 [28].

As in the case of equation (3.1.1), no exact solution seems to exist, and so the same Runge-Kutta solution [5] was used as a basis for comparison. The same values for a and b were used, while the values for $\frac{q}{M}$ were 0.0, 0.1, and 0.25, restricting the investigation to the under-damped case. The same initial conditions were used, with X_0 equal to 1.0.

B. Derivation of Piecewise Linear Solution

The derivation for the equation of the linear segments is exactly the same as was described in the previous chapters, using essentially the same method to find the necessary factors such as x_0^* and p_0^* . A slight modification in the procedure to determine the damping constant $\frac{c}{M}$

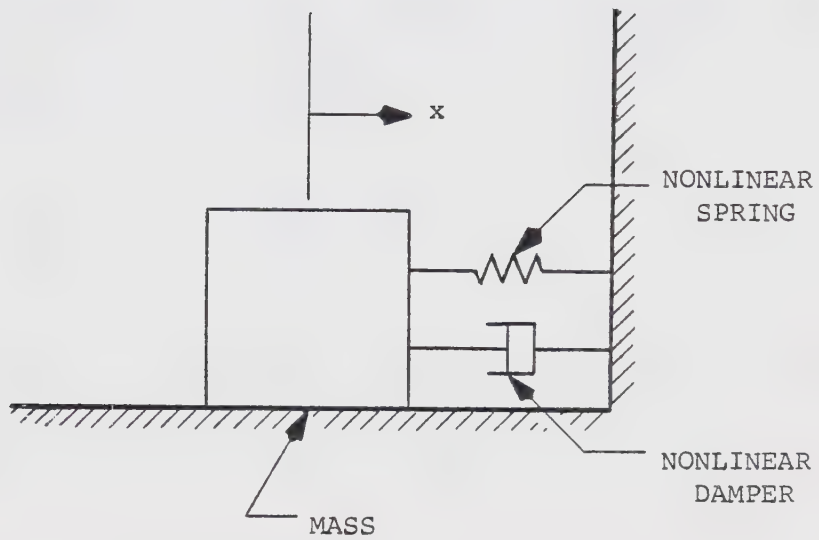


FIGURE 4.1 - MASS-SPRING SYSTEM (NONLINEAR DAMPING) [28]

is required as it changes in value from one segment to another. Since it is desired to obtain a solution whose form is similar to that of (1.2), the expression for the damping as given in (4.1.1) is replaced by an equivalent one resembling what was seen in (3.1.1), or, in other words, a linear approximation over a small increment of displacement

The starting point for this is part of the problem. One possibility involves an iteration within an iteration in order to obtain an average value of the velocity, and thus the damping force. This is done as follows.

At the beginning of motion, no damping is assumed, and the form of the equation is taken to be [19]:

$$\frac{d^2}{dt^2}(x - x_0^*) + \frac{k_0}{M}(x - x_0^*) = 0 .$$

From this, an end velocity can be found. This is averaged, and a damping coefficient is determined using:

$$c = \frac{q |\dot{x}_{AV}|^2 \operatorname{sgn}(\dot{x}_{AV})}{\dot{x}_{AV}} ,$$

\dot{x}_{AV} = Average segment velocity.

The iteration is done again, with the same initial conditions, but the equation now becomes [19], [29]:

$$\frac{d^2}{dt^2}(x - x_0^*) + \frac{c}{M} \dot{x} + \frac{k_0}{M}(x - x_0^*) = 0 .$$

Now the current velocity is calculated and compared with the previous

final value. If their difference is greater than a given tolerance, the iteration starts again, with a new damping coefficient calculated from the average segment velocity, based upon this new final speed. Once this tolerance is achieved, the end conditions become the initial conditions for the next segment, and the entire procedure starts again as before, except now for all subsequent segments, the new initial velocity is used each time.

Since the derivation is the same as before, the solution will essentially be identical to that of the previous chapter [19], [29] or:

$$x = x_0^* + e^{-n(t - t_0)} \left[A \cos p^*(t - t_0) + B \sin p^*(t - t_0) \right], \quad (4.2.1)$$

$$\dot{x} = e^{-n(t - t_0)} \left[(-nA + p^*B) \cos p^*(t - t_0) - (p^*A + nB) \sin p^*(t - t_0) \right], \quad (4.2.2)$$

where the coefficients were determined as before, and are:

$$A = x_0 - x_0^*,$$

$$B = \frac{\dot{x}_0 + n(x_0 - x_0^*)}{p^*},$$

$$n = \frac{c}{2M}.$$

The values for $t_1 - t_0$ were calculated using the Newton-Raphson method [30], and are subject to the conditions described in the previous chapter, including the procedure for finding the point at which the system bottoms out in the first trough.

C. Results

The equations for the piecewise linear solution and the required commands to use DVERK [5] (including parameters and the calling statement) were assembled into a computer program, and solution plots generated by running them on the Amdahl 470V/8 [26] mentioned earlier. The size of the displacement increment used by the piecewise linear method was the same as the one used in the previous chapter ($\Delta x = 0.1$), as well as the same size of time increment for the Runge-Kutta method [5] ($\Delta t = 0.05$).

The results of the computer plot program that calculated both solutions can be seen in Figures 4.2 - 4.7. Once again, one can see that a good agreement was achieved, even when using a fairly large value for Δx , and it would seem from the output that the piecewise linear method successfully calculated the location of the first trough. It can be concluded from this that the parameter values and the size of the displacement interval were sufficient to give satisfactory results for the given conditions.

The program was stripped to the essential steps that contained only the calculations and the execution times were determined for it [27], which should give one a reasonable estimate as to the relative speeds (and hence, the costs) of the two methods. As in the previous chapter, the basis of judgement was the number of divisions of the first quarter-cycle. This was because the piecewise linear solution used an increment of displacement, while the Runge-Kutta method [5] required increments of time.

The values obtained are to be found on Figures 4.8 - 4.10, and were taken from tests conducted on the Amdahl 470V/7 [25], using an older and

FIGURE 4.2 - SOLUTIONS TO HARD SPRING EQUATION
(NONLINEAR DAMPING) FOR $a = 1.0$, $b = 0.15$, $\frac{g}{M} = 0.0$,
 $\Delta x(0) = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0)^M = 0.0$

(N.B.: Runge-Kutta method uses [5].)

$a = 1.0000$ $b = 0.1500$
 $q/M = 0.0000$

Δx (PIECEWISE
 LINEARIZATION) = 0.1000

Δt (RUNGE-KUTTA) = 0.0500

$x(0) = 1.0000$ $\dot{x}(0) = 0.0000$

— RUNGE-KUTTA

⊙ PIECEWISE LINEARIZATION

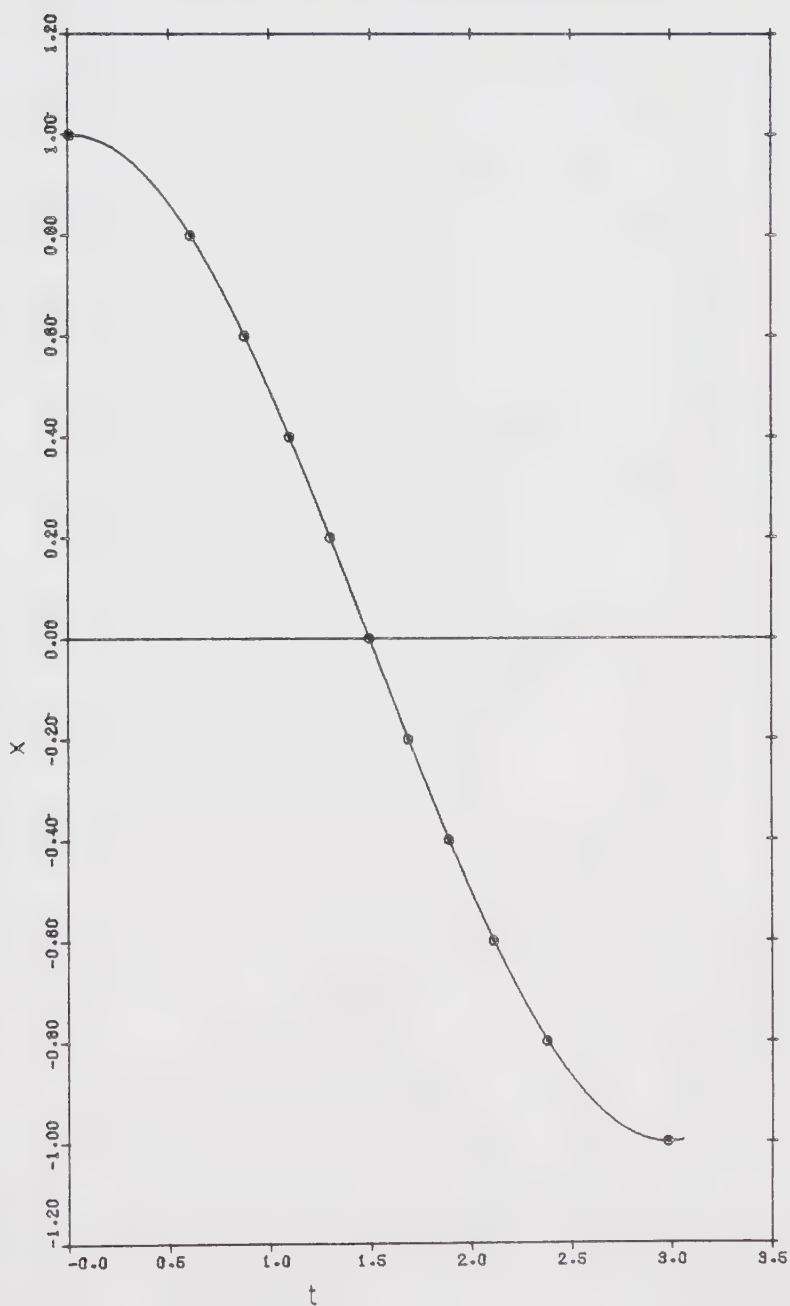


FIGURE 4.3 - SOLUTIONS TO HARD SPRING EQUATION
(NONLINEAR DAMPING) FOR $a = 1.0$, $b = 2.0$, $\frac{c}{M} = 0.0$,
 $\Delta x(0) = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta method uses [5].)

$a = 1.0000$ $b = 2.0000$
 $q/M = 0.0000$

Δx (PIECEWISE
 LINEARIZATION) = 0.1000

Δt (RUNGE-KUTTA) = 0.0500

$x(0) = 1.0000$ $\dot{x}(0) = 0.0000$

— RUNGE-KUTTA

○ PIECEWISE LINEARIZATION

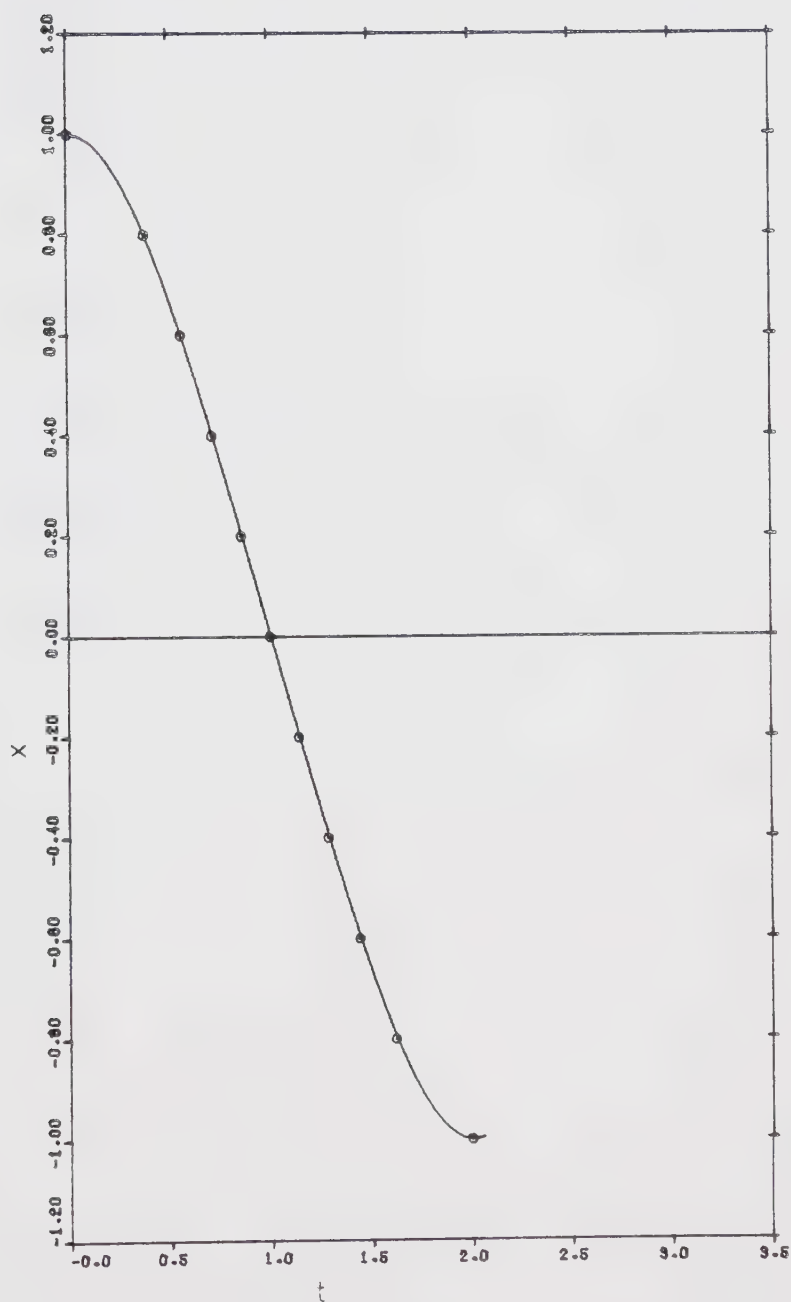


FIGURE 4.4 - SOLUTIONS TO HARD SPRING EQUATION
(NONLINEAR DAMPING) FOR $a = 1.0$, $b = 0.15$, $\frac{g}{M} = 0.10$,
 $\Delta x(0) = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0)^M = 0.0$

(N.B.: Runge-Kutta method uses [5].)

$$a = 1.0000 \quad b = 0.1500$$

$$q/M = 0.1000$$

$$\Delta x \text{ (PIECEWISE LINEARIZATION)} = 0.1000$$

$$\Delta t \text{ (RUNGE-KUTTA)} = 0.0500$$

$$x(0) = 1.0000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

○ PIECEWISE LINEARIZATION

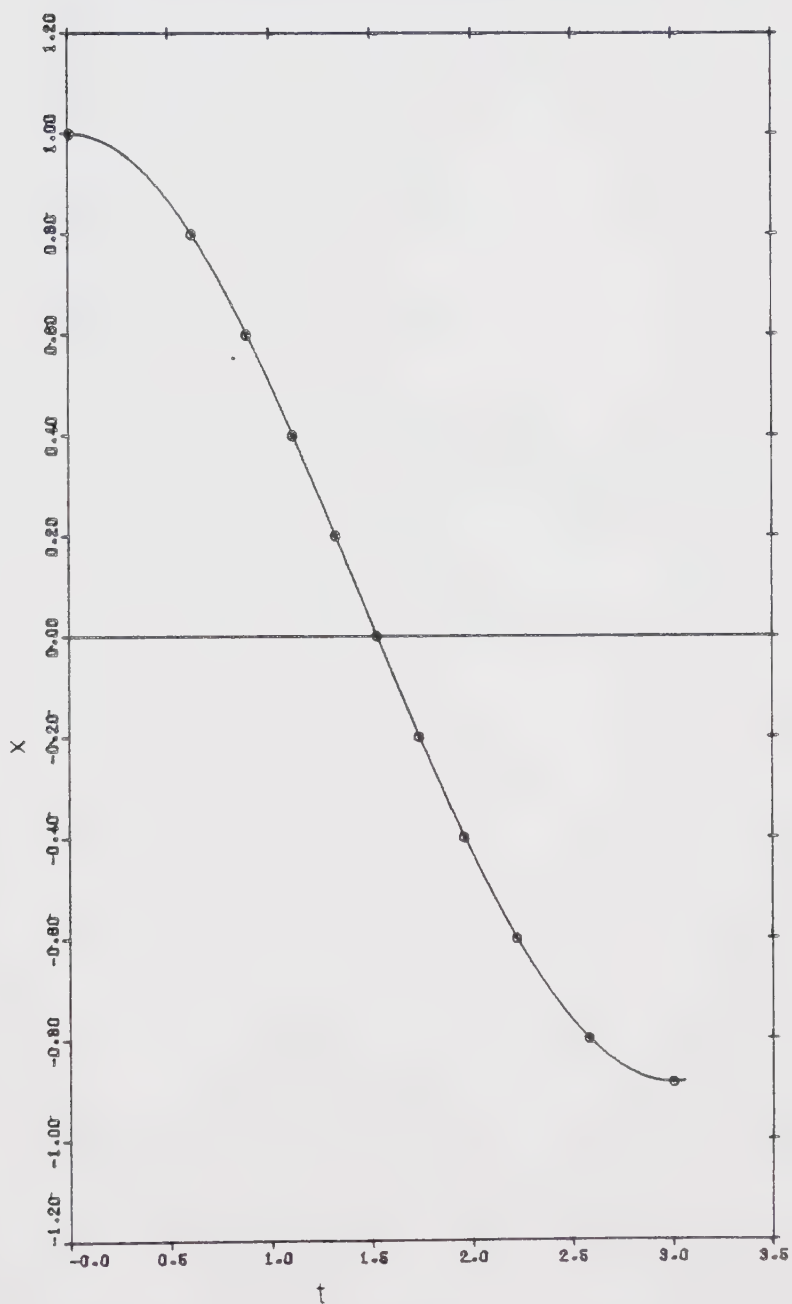


FIGURE 4.5 - SOLUTIONS TO HARD SPRING EQUATION
(NONLINEAR DAMPING) FOR $a = 1.0$, $b = 2.0$, $\frac{g}{M} = 0.10$,
 $\Delta x(0) = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta method uses [5].)

$a = 1.0000$ $b = 2.0000$
 $q/M = 0.1000$

Δx (PIECEWISE
 LINEARIZATION) = 0.1000

Δt (RUNGE-KUTTA) = 0.0500

$x(0) = 1.0000$ $\dot{x}(0) = 0.0000$

— RUNGE-KUTTA

⊙ PIECEWISE LINEARIZATION

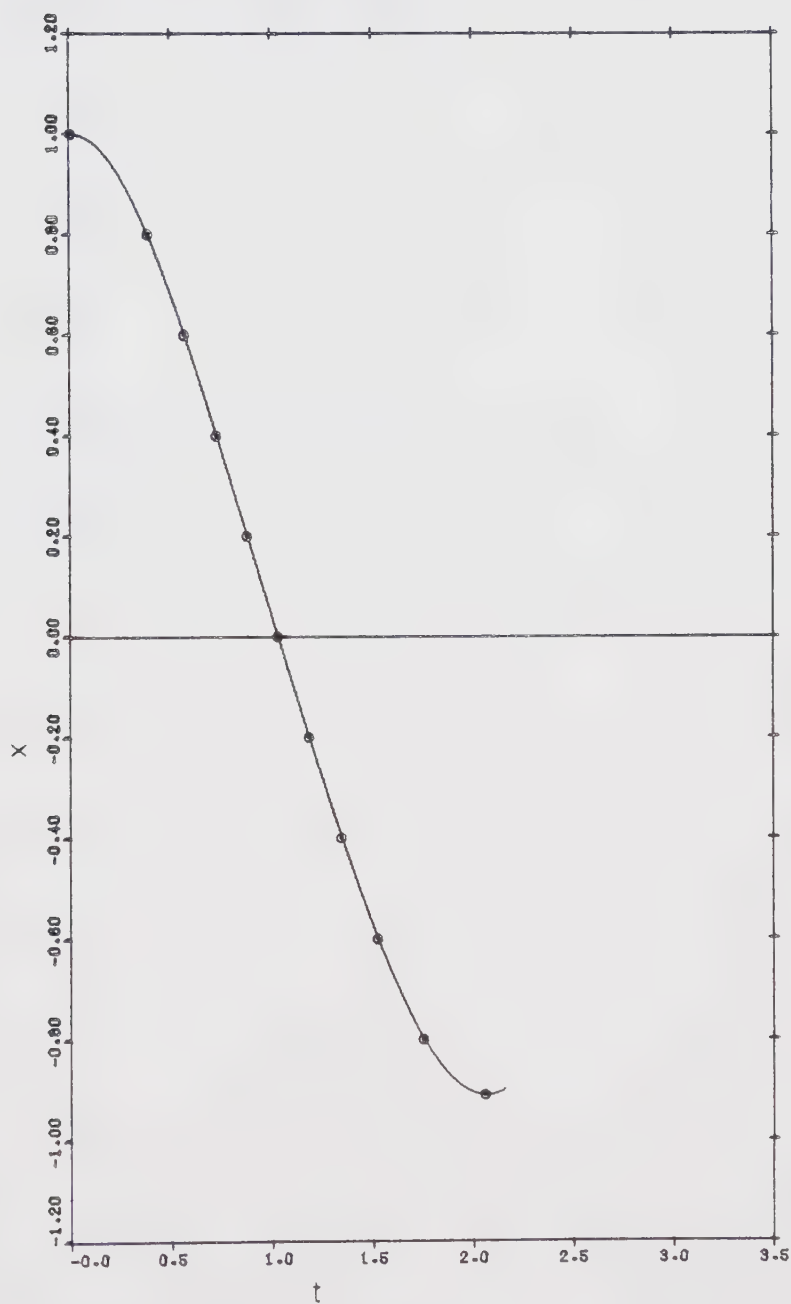


FIGURE 4.6 - SOLUTIONS TO HARD SPRING EQUATION
(NONLINEAR DAMPING) FOR $a = 1.0$, $b = 0.15$, $\frac{g}{M} = 0.25$,
 $\Delta x(0) = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta method uses [5].)

$$a = 1.0000 \quad b = 0.1500$$

$$q/M = 0.2500$$

$$\Delta x \text{ (PIECEWISE LINEARIZATION)} = 0.1000$$

$$\Delta t \text{ (RUNGE-KUTTA)} = 0.0500$$

$$x(0) = 1.0000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

○ PIECEWISE LINEARIZATION

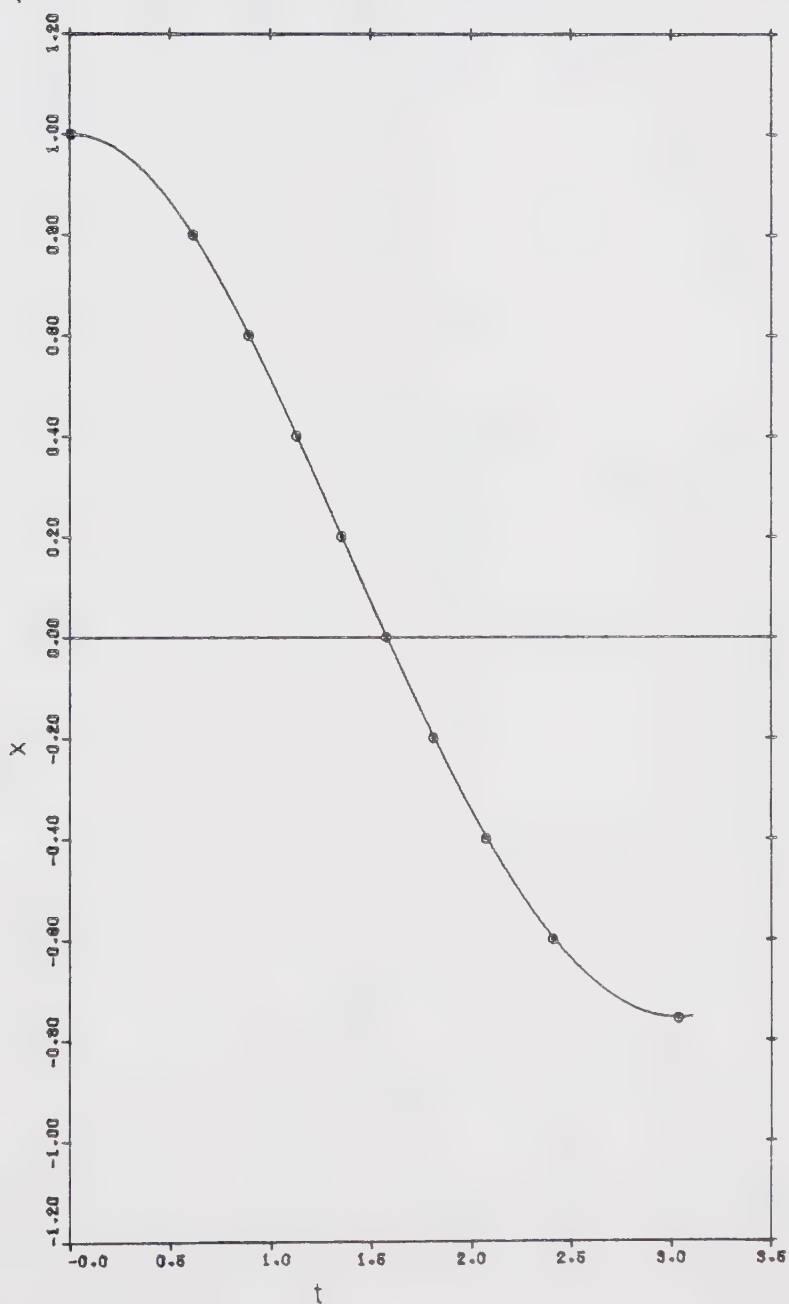


FIGURE 4.7 - SOLUTIONS TO HARD SPRING EQUATION
(NONLINEAR DAMPING) FOR $a = 1.0$, $b = 2.0$, $\frac{g}{M} = 0.25$,
 $\Delta x(0) = 0.10$, $\Delta t = 0.05$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta method uses [5].)

$a = 1.0000$ $b = 2.0000$
 $q/M = 0.2500$

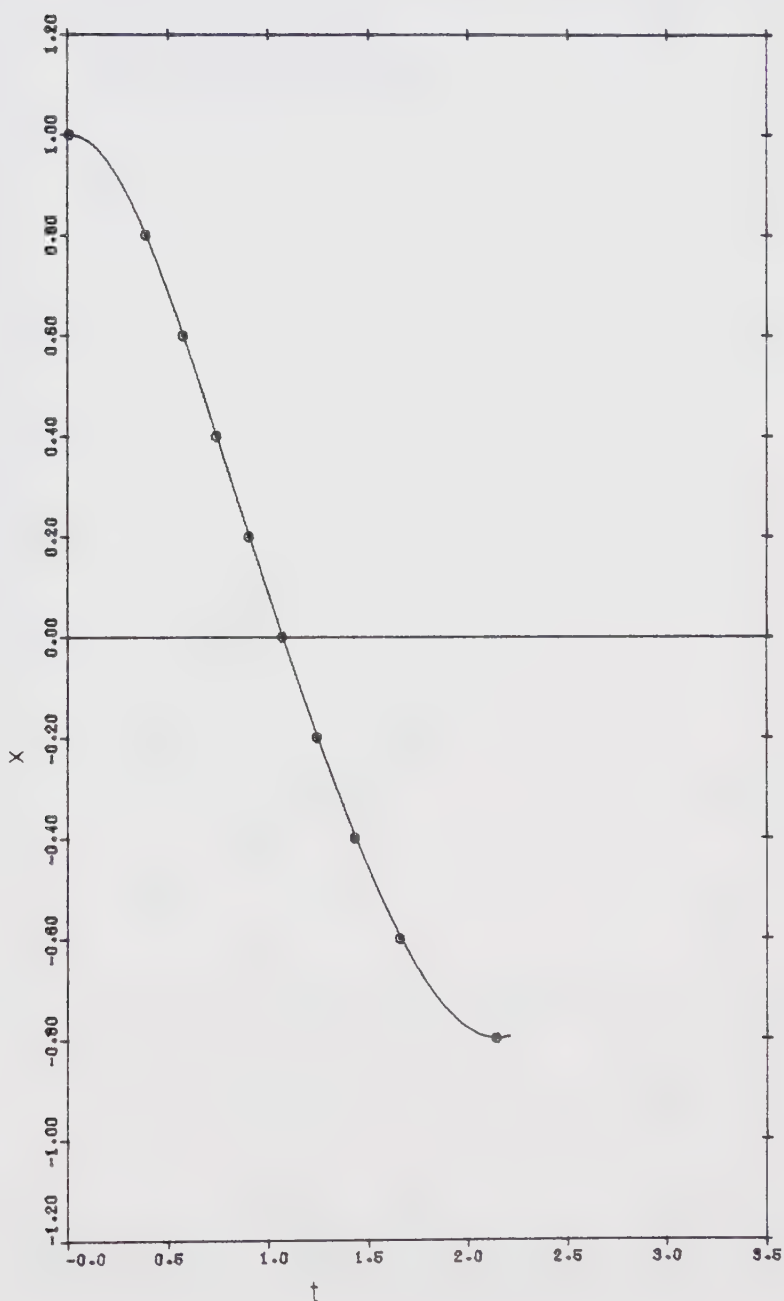
Δx (PIECEWISE
 LINEARIZATION) = 0.1000

Δt (RUNGE-KUTTA) = 0.0500

$x(0) = 1.0000$ $\dot{x}(0) = 0.0000$

— RUNGE-KUTTA

○ PIECEWISE LINEARIZATION



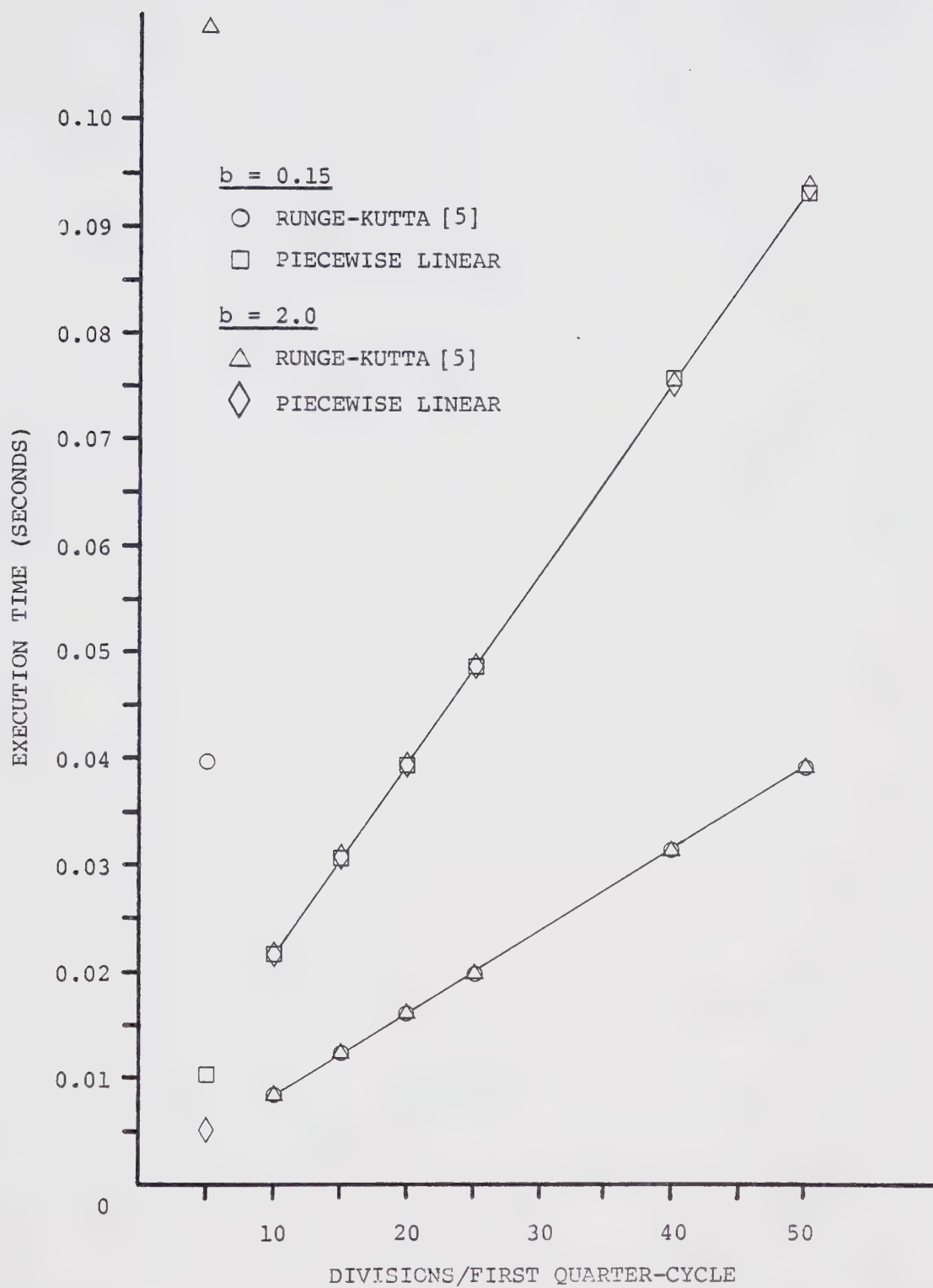


FIGURE 4.8 - EXECUTION TIMES FOR SOLUTIONS TO HARD SPRING EQUATION (NONLINEAR DAMPING) FOR $a = 1.0$, $\frac{g}{M} = 0.0$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$ [27]

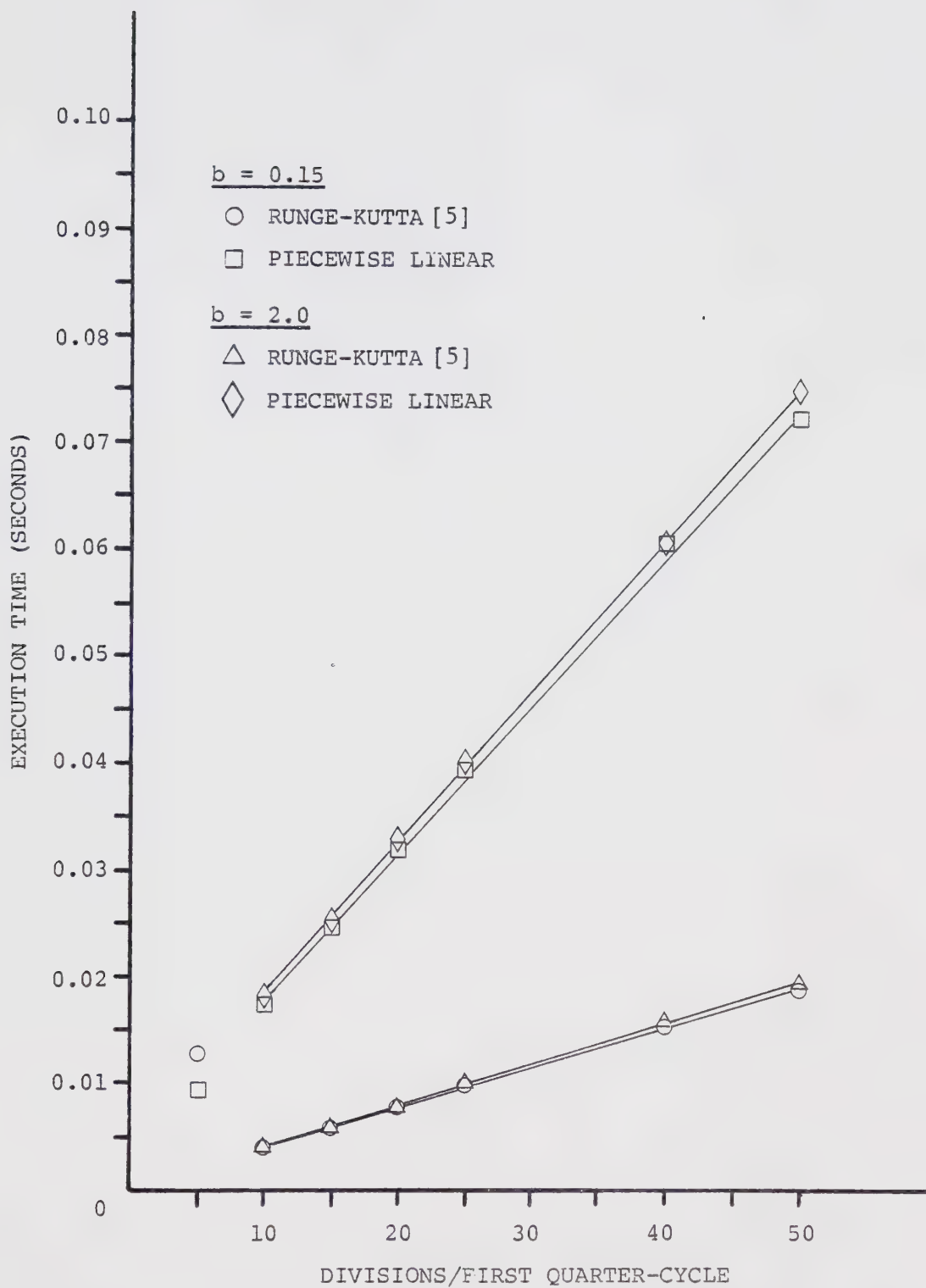


FIGURE 4.9 - EXECUTION TIMES FOR SOLUTIONS TO HARD SPRING EQUATION (NONLINEAR DAMPING) FOR $a = 1.0$, $\frac{q}{M} = 0.10$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$ [27]

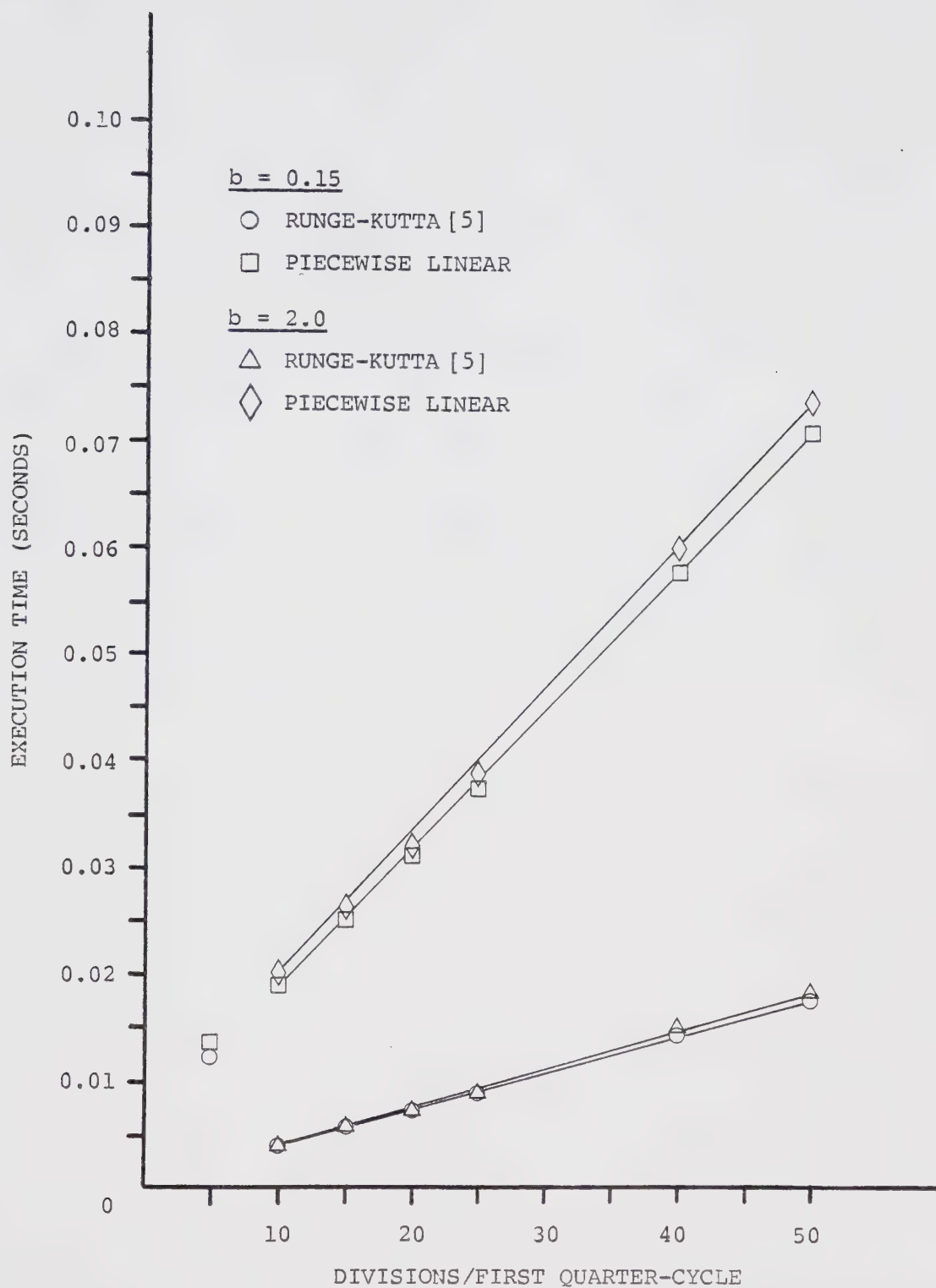


FIGURE 4.10 - EXECUTION TIMES FOR SOLUTIONS TO HARD SPRING EQUATION (NONLINEAR DAMPING) FOR $a = 1.0$, $\frac{g}{M} = 0.25$, $x(0) = 1.0$, $\dot{x}(0) = 0.0$ [27]

slower version of the program.

Once again, it was found that the time required was linear with respect to the number of divisions per first quarter-cycle. As before, for 5 divisions of the first quarter-cycle, problems were experienced in obtaining a reasonable execution time for either result, depending upon the value of b and $\frac{q}{M}$ (since an increase in either or both caused the program to exceed its pre-set execution time limit [31]). This leads to the same conclusion as before that there may be a lower limit to the number of points that can be used for either method to be effective or economical. One of the things that was surprising was the lowering of the execution time for the piecewise method for $a = 1.0$, $b = 2.0$, and $\frac{q}{M} = 0.0$. This could be regarded as coincidence, unless there was some factor in the computer system itself that may have led to this which is not immediately apparent.

V. VAN DER POL EQUATION

A. Preliminary Comments

The methods of solution described in the previous two chapters gave stable results since, when the damping factors were varied over a large range for the underdamped case, no erratic behaviour or improbable wave forms were encountered. In this chapter, however, an equation will be examined that is different in that the term representing its resisting force varies with respect to both the displacement and the velocity, its solution curve varies radically as a parameter is varied over a wide range, and its phase trajectory exhibits a limit cycle. This is Van der Pol's equation [32]:

$$\ddot{x} + \mu(x^2 - 1)\dot{x} + x = 0 . \quad (5.1.1)$$

The solutions to the linear equations are essentially the same as were found in the previous chapters, except that instead of being restricted to only the underdamped situation, the critically damped and overdamped cases are solved for as well, owing to the nature of the solution to (5.1.1).

Owing to the wide variation in behaviour that this equation can exhibit, the initial conditions will be the same as had been previously used with the exception that X_0 will vary in value to allow investigation inside, on, and outside the limit cycle. The parameter μ will have the values 0.25, 1.0, 2.0, and 5.0 so that the variations in the solution curves can be examined.

While the previous systems had nonlinear restoring forces, (5.1.1)

does not, and so, this will mean that many of the calculations that were needed in order to set up the solution to the linear equation are not needed in this situation. This also means that an interval of time can be used instead of displacement with resulting simplification in the calculations.

Since an exact solution does not seem to exist, the same Runge-Kutta solution [5] that had been used in the previous two chapters will be the benchmark against which the piecewise linearization will be compared.

B. Derivation of Piecewise Linear Solution

Equation (5.1.1) can be approximated over a small increment of time by the equation [29]:

$$\ddot{x} + 2n\dot{x} + p^2x = 0 , \quad (5.2.1)$$

where:

$$n \approx \frac{u(x^2 - 1)}{2} , \quad (5.2.2)$$

$$p^2 = 1.0 .$$

The solutions of (5.2.1) can be found in texts (for example, Timoshenko, et. al. [29]), and are as follows.

For $p^2 - n^2 > 0$ [35], [36]:

$$x = e^{-nt} \left[x_0 \cos p^*t + \left(\frac{nx_0 + \dot{x}_0}{p^*} \right) \sin p^*t \right] , \quad (5.2.3)$$

$$\dot{x} = e^{-nt} \left\{ \dot{x}_0 \cos p^* t + \left[\frac{-n(nx_0 + \dot{x}_0)}{p^*} - p^* x_0 \right] \sin p^* t \right\} , \quad (5.2.4)$$

$$p^* = \sqrt{p^2 - n^2} .$$

For $p^2 - n^2 = 0$ [36], [37]:

$$x = e^{-pt} [x_0 + (nx_0 + \dot{x}_0)t] , \quad (5.2.5)$$

$$\dot{x} = e^{-pt} [-px_0 + (nx_0 + \dot{x}_0)(1 - pt)] . \quad (5.2.6)$$

For $p^2 - n^2 < 0$ [36], [37]:

$$x = e^{-nt} \left[x_0 \cosh (p^{**}t) + \left(\frac{nx_0 + \dot{x}_0}{p^{**}} \right) \sinh (p^{**}t) \right] , \quad (5.2.7)$$

$$\dot{x} = e^{-nt} \left\{ \dot{x}_0 \cosh (p^{**}t) + \left[\frac{-n(nx_0 + \dot{x}_0)}{p^{**}} + p^{**}x_0 \right] \sinh (p^{**}t) \right\} , \quad (5.2.8)$$

$$p^{**} = \sqrt{n^2 - p^2} .$$

The major problem here is how to estimate n . Assume at the beginning of a segment some initial conditions of x_0 and \dot{x}_0 , and a time interval of Δt_p . An initial guess would use x_0 for x in (5.2.2) and an initial estimate of the change in displacement is calculated (which will

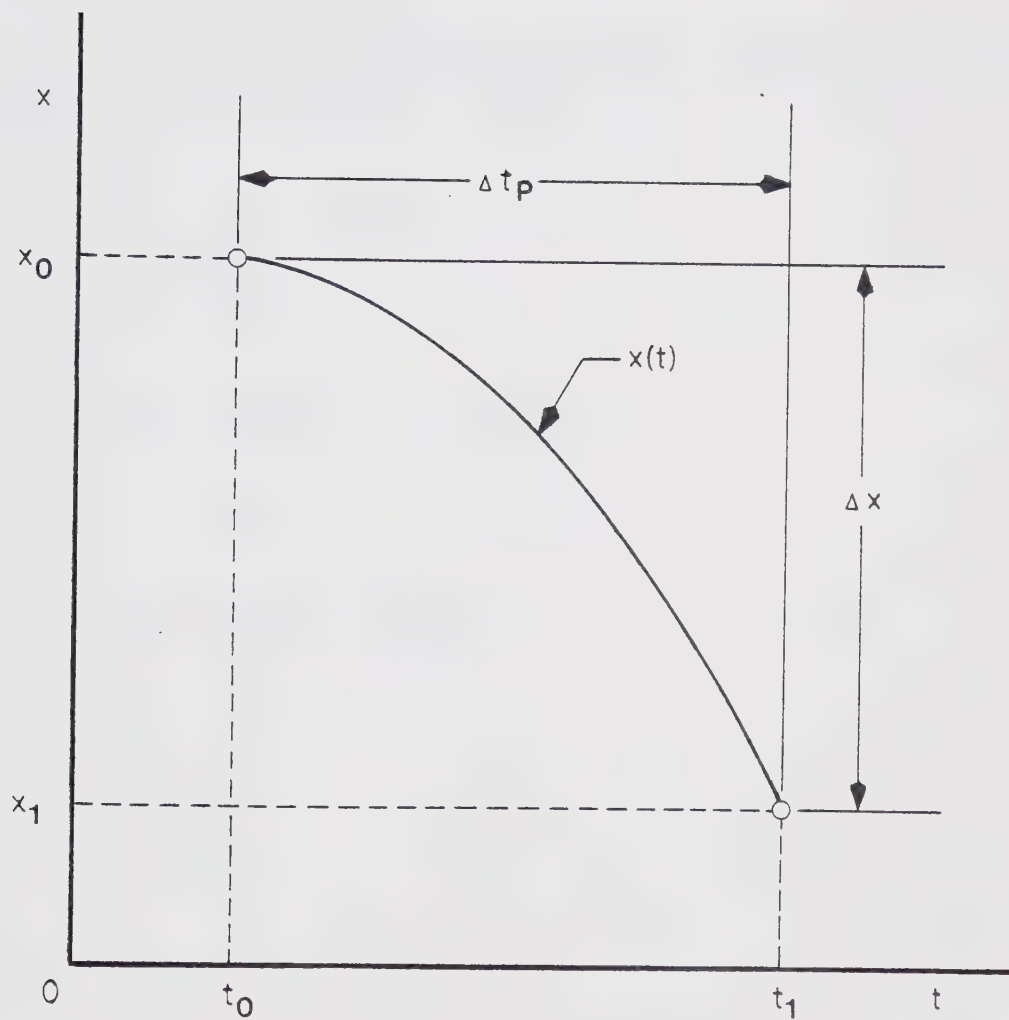


FIGURE 5.1 - TYPICAL INTERVAL LAYOUT (VAN DER POL EQUATION)

be explained further on). The new displacement x_1 is $x_0 + \Delta x$, and this is compared to a previously calculated value of x_1 (x_{PREV}). Should the difference between x_1 and x_{PREV} be very small, x_1 is the final displacement at the end of Δt_p . If not, x_1 becomes x_{PREV} , x now is $x_0 + \Delta x/2$, and a new n is found, as well as new values for Δx and x . Figure 5.1 should clarify matters.

This is how the solution is carried out in general for each interval. At the beginning of the iteration,

t_0 = Initial time,

$t_0 + \Delta t_p$ = Final time

= t_1 ,

x_0 = Initial displacement

$x_0 + \Delta x$ = Final displacement

= x_1 ,

\dot{x}_0 = Initial velocity,

\dot{x}_1 = Final velocity.

Once the displacement approaches either a peak or trough, a variation in this pattern is introduced to find the time and displacement at these points. (See Figure 5.2 for details.) The procedure is as follows. Starting at x_0 and t_0 , x_1 is found as before. The value of $\text{sgn}(\dot{x}_1)$ is calculated, and if found to be different from $\text{sgn}(\dot{x}_0)$, Δt_1 (the time taken to reach x_{11}) is found by determining the time interval needed for \dot{x} to reach zero using a Newton-Raphson method [30], starting at x_0 . This assumes that \dot{x} not zero at the end of a complete segment. Should it be, this section is circumvented.

The value for x_{11} is found by iteration much like the bottom of the

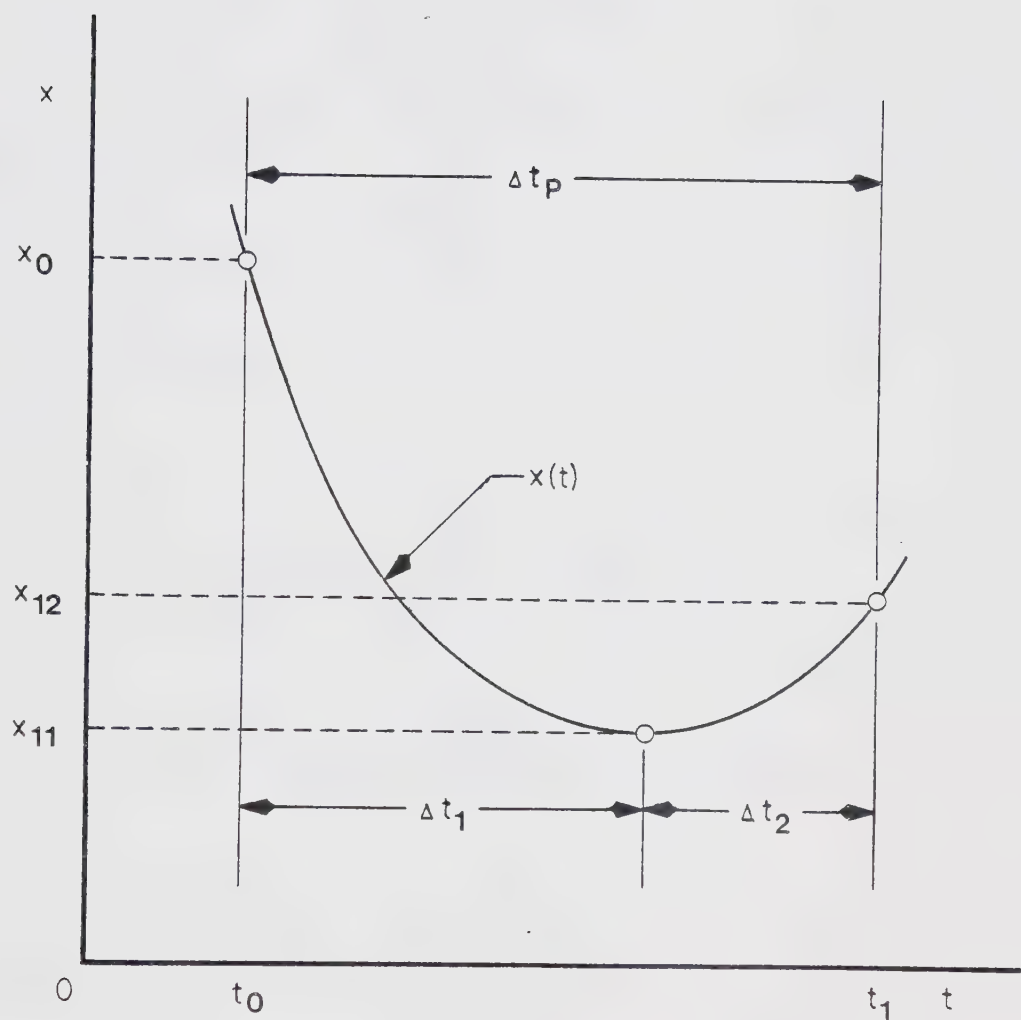


FIGURE 5.2 - LAYOUT OF PEAK OR TROUGH (VAN DER POL EQUATION)

first trough was determined in the previous two chapters. The estimated value is x_F , with the first guess being x_{12} . Then, using Δt_1 , a change in displacement is found (a new Δx). The value of $(x_0 + \Delta x) - x_F$ is determined, and if very small, x_{11} is equal to $x_0 + \Delta x$. If not, $x_0 + \Delta x$ becomes x_F and the iteration starts again.

This completes the first half-cycle.

Since the peak or trough is most likely not going to occur at the end of a time interval, and it would be convenient to find the displacement at the end of Δt_p (in order to continue afterwards with even time intervals until the next peak or trough), x_{12} is found based upon Δt_p , with the remainder of the half-cycle as was previously done. The displacement-time curve and phase-plane diagram can now be obtained.

It is necessary to go through this procedure as one never knows in advance how long the system takes to reach the limit cycle, since it depends upon the value of γ as well as the initial conditions. So, for convenience, the number of half-cycles that had been calculated will be monitored and the calculation stops once the required number had been achieved.

When the displacement approaches a peak or trough, Δt_1 is determined using the Newton-Raphson method [30]:

$$(\Delta t)_{i+1} = (\Delta t)_i - \frac{\dot{x}[(\Delta t)_i]}{\ddot{x}[(\Delta t)_i]}$$

where $\dot{x}[(\Delta t)_i]$ is the expression for the velocity (which particular one would depend upon what value $p^2 - n^2$ has), and $\ddot{x}[(\Delta t)_i]$ is the acceleration for that particular situation.

The following approximations to the velocities and accelerations are used, based upon Timoshenko, et. al. [34].

For $p^2 - n^2 > 0$ [35], [36]:

$$\begin{aligned} \dot{x} [(\Delta t)_i] &= e^{-n(\Delta t)_i} \{A \cos [p^*(\Delta t)_i] \\ &+ B \sin [p^*(\Delta t)_i]\} , \end{aligned} \quad (5.2.9)$$

$$\begin{aligned} \ddot{x} [(\Delta t)_i] &= e^{-n(\Delta t)_i} \{(-nA + p^*B) \\ &\cdot \cos [p^*(\Delta t)_i] \\ &+ (-p^*A - nB) \sin [p^*(\Delta t)_i]\} , \end{aligned} \quad (5.2.10)$$

$$A = \dot{x}_0,$$

$$B = -\frac{n(nx_0 + \dot{x}_0)}{p^*} - p^*x_0.$$

For $p^2 - n^2 = 0$ [35], [36]:

$$\begin{aligned} \dot{x} [(\Delta t)_i] &= e^{-p(\Delta t)_i} \{-px_0 + (nx_0 + \dot{x}_0) \\ &\cdot [1 - p(\Delta t)_i]\} , \end{aligned} \quad (5.2.11)$$

$$\begin{aligned} \ddot{x} [(\Delta t)_i] &= e^{-p(\Delta t)_i} \{p^2x_0 - p(nx_0 + \dot{x}_0) \\ &\cdot [2 - p(\Delta t)_i]\} . \end{aligned} \quad (5.2.12)$$

For $p^2 - n^2 < 0$ [35], [36]:

$$\begin{aligned} \dot{x} [(\Delta t)_i] &= e^{-n(\Delta t)_i} \{C \cosh [p^{**}(\Delta t)_i] \\ &+ D \sinh [p^{**}(\Delta t)_i]\} , \end{aligned} \quad (5.2.13)$$

$$\begin{aligned} \ddot{x}[(\Delta t)_i] = & e^{-n(\Delta t)_i} \{ (-nC + p^{**}D) \\ & \cdot \cosh[p^{**}(\Delta t)_i] \\ & + (p^{**}C - nD) \sinh[p^{**}(\Delta t)_i] \} , \end{aligned} \quad (5.2.14)$$

$$C = \dot{x}_0,$$

$$D = - \frac{n(nx_0 + \dot{x}_0)}{p^{**}} + p^{**}x_0 .$$

During these calculations, the value of n is found, and this ultimately determines what the solution for the segment will be. The change in displacement, Δx , is found by finding $(x_1 - x_0)$, corresponding to the appropriate value of n . Since equal time intervals are used, Δt_p is used instead of t in equations (5.2.3) - (5.2.8), where $\Delta t_p = t_1 - t_0$ (based on the functions being dependent upon $t - t_0$ instead of t , due to the piecewise linearization).

C. Results

Equations (5.2.3) - (5.2.14) were assembled into a computer program, as well as a Runge-Kutta solution using DVERK [5], and the results obtained by running it on the Amdahl 470 V/8 [26].

Initially, the approximation was tested by using a small value for μ as a check that it worked. This was arbitrarily chosen to be 0.25 and was tested for X_0 being equal to 1.0, 2.0, and 2.5 with Δt_p for the piecewise linearization being equal to 0.2, and the Δt_{RK} for the Runge-Kutta solution [5] being 0.05. The size of the time intervals was chosen arbitrarily. The results are seen on Figures 5.3 - 5.8. (All solutions and phase trajectories in this chapter are based on [38],

FIGURE 5.3 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 0.25, \Delta t_p = 0.20, \Delta t_{RK} = 0.05, x(0) = 1.0,$
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)

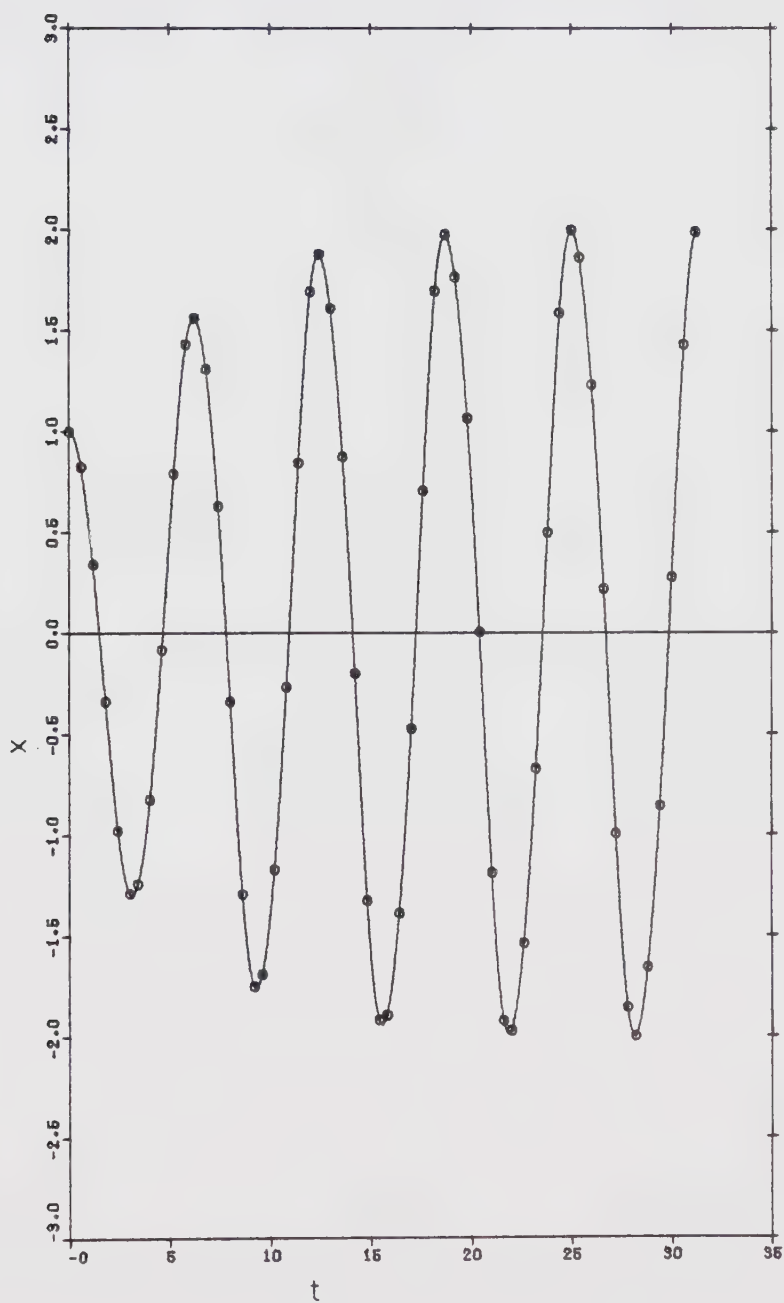
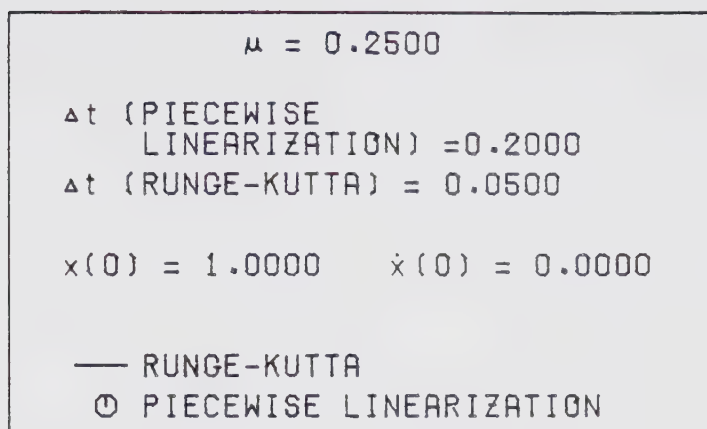


FIGURE 5.4 - PHASE TRAJECTORIES FOR VAN DER POL
EQUATION FOR $\mu = 0.25$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$,
 $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [39].)

$$\mu = 0.2500$$

Δt (PIECEWISE
LINEARIZATION) = 0.2000

Δt (RUNGE-KUTTA) = 0.0500

$$x(0) = 1.0000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

○ PIECEWISE LINEARIZATION

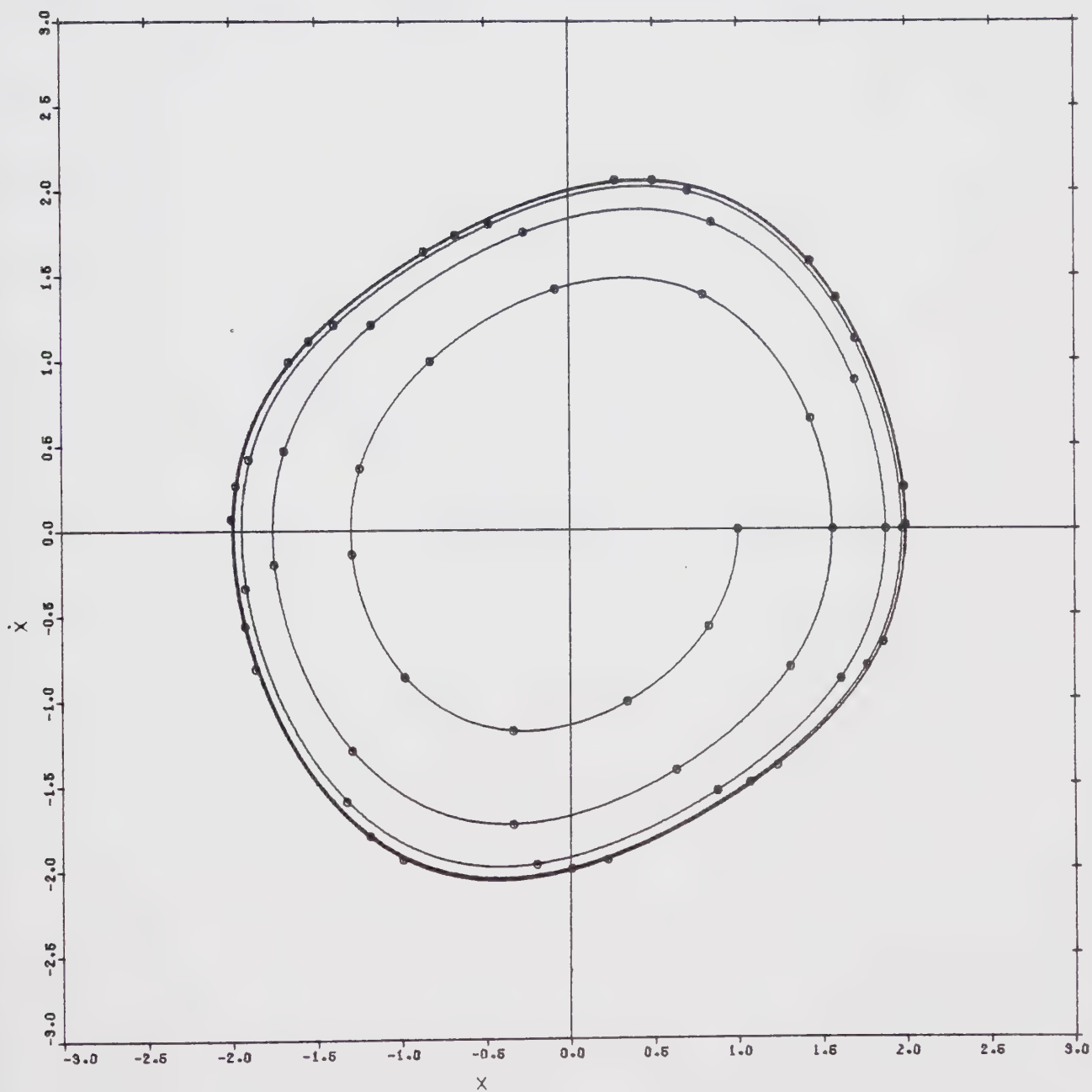


FIGURE 5.5 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 0.25, \Delta t_p = 0.20, \Delta t_{RK} = 0.05, x(0) = 2.0,$
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)

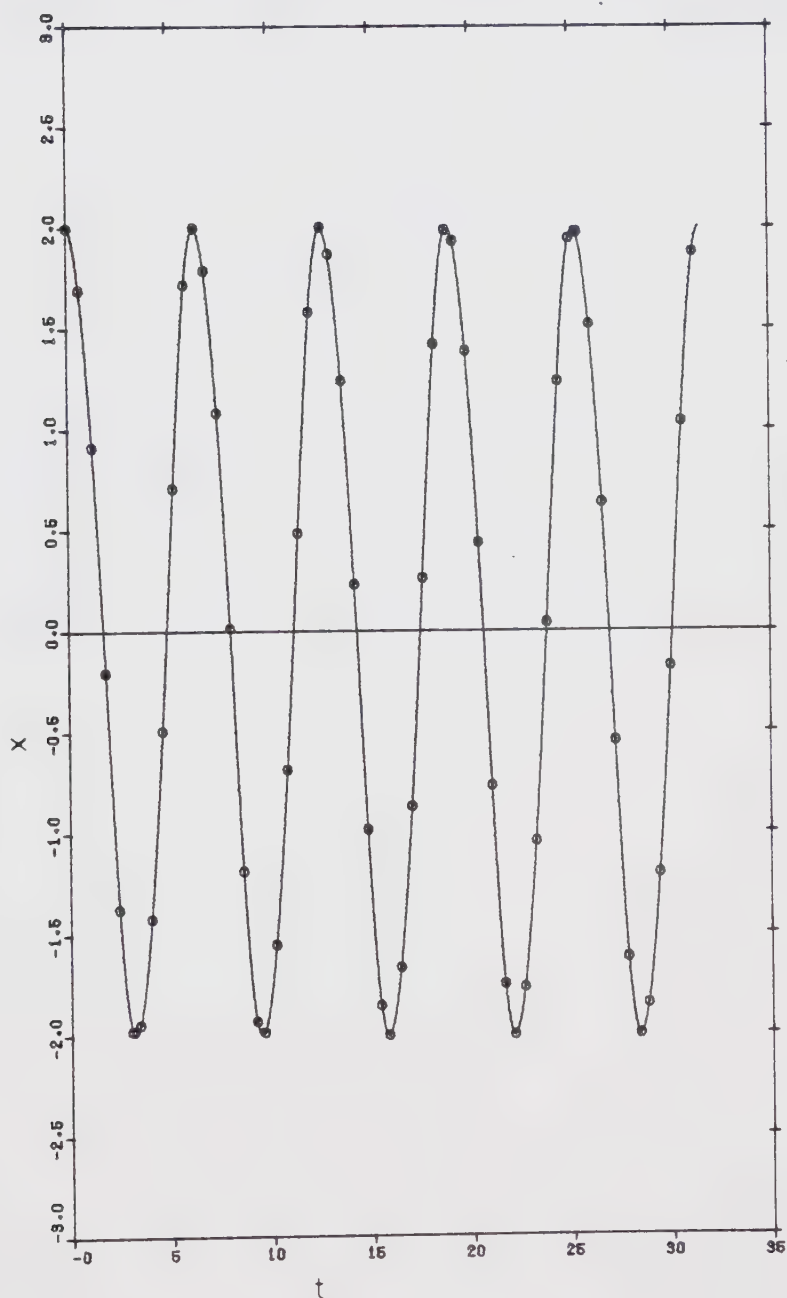
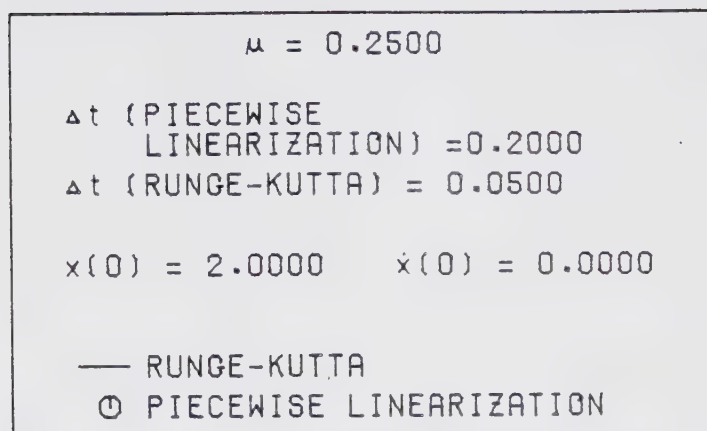


FIGURE 5.6 - PHASE TRAJECTORIES FOR VAN DER POL
EQUATION FOR $\mu = 0.25$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$,
 $x(0) = 2.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [39].)

$$\mu = 0.2500$$

Δt (PIECEWISE
LINEARIZATION) = 0.2000

Δt (RUNGE-KUTTA) = 0.0500

$$x(0) = 2.0000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

○ PIECEWISE LINEARIZATION

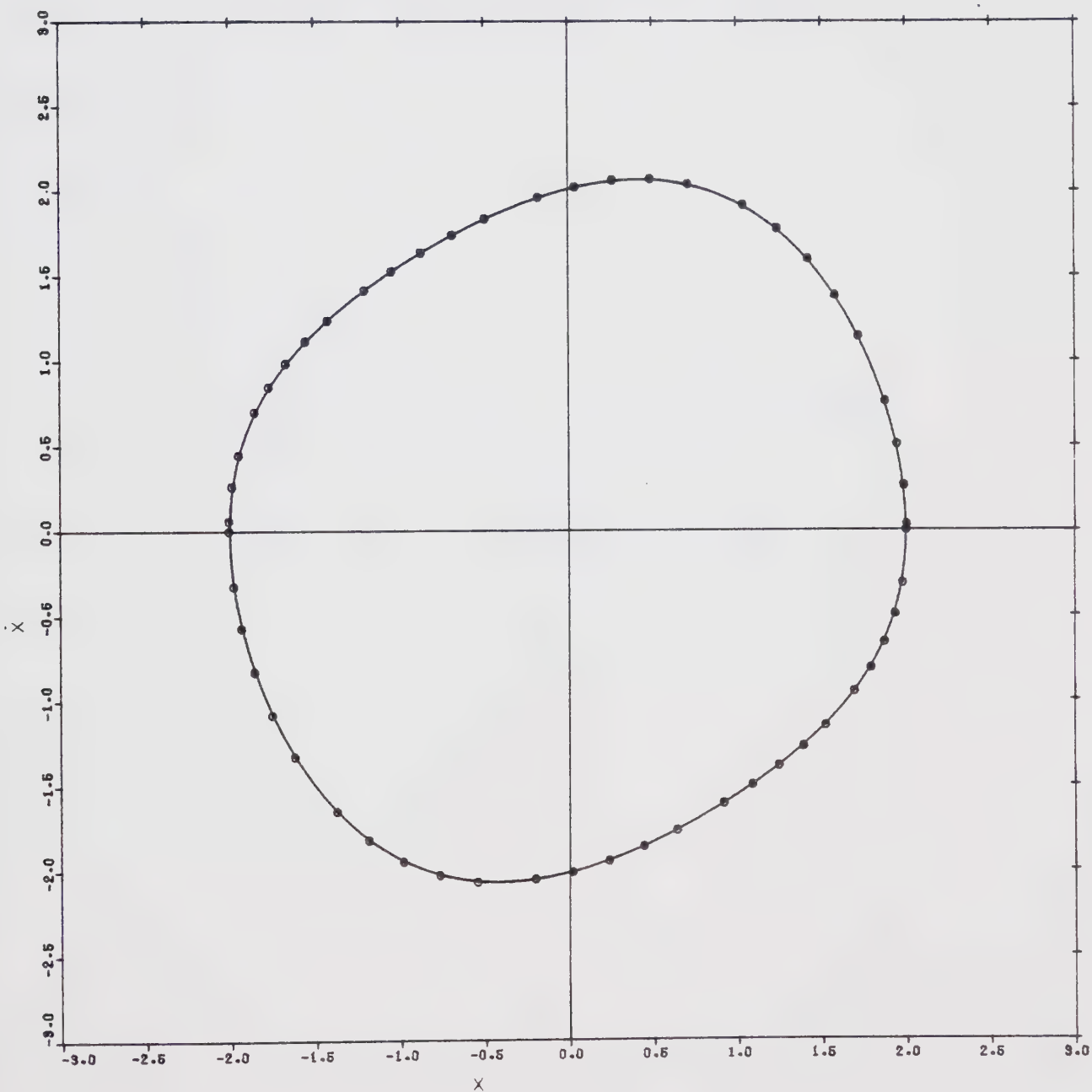


FIGURE 5.7 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 0.25, \Delta t_p = 0.20, \Delta t_{RK} = 0.05, x(0) = 2.5,$
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)

$$\mu = 0.2500$$

Δt (PIECEWISE
LINEARIZATION) = 0.2000

Δt (RUNGE-KUTTA) = 0.0500

$$x(0) = 2.5000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

○ PIECEWISE LINEARIZATION

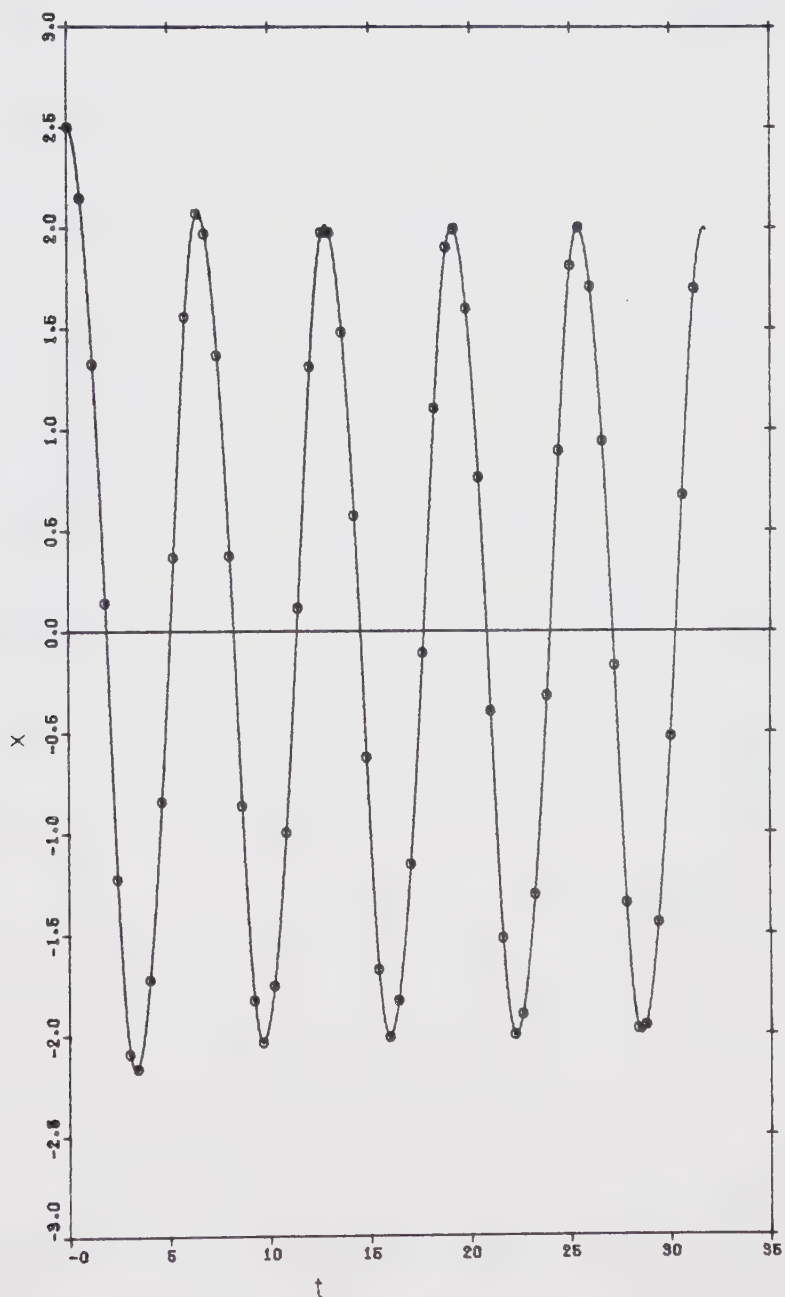


FIGURE 5.8 - PHASE TRAJECTORIES FOR VAN DER POL
EQUATION FOR $\mu = 0.25$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$,
 $x(0) = 2.5$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [39].)

$$\mu = 0.2500$$

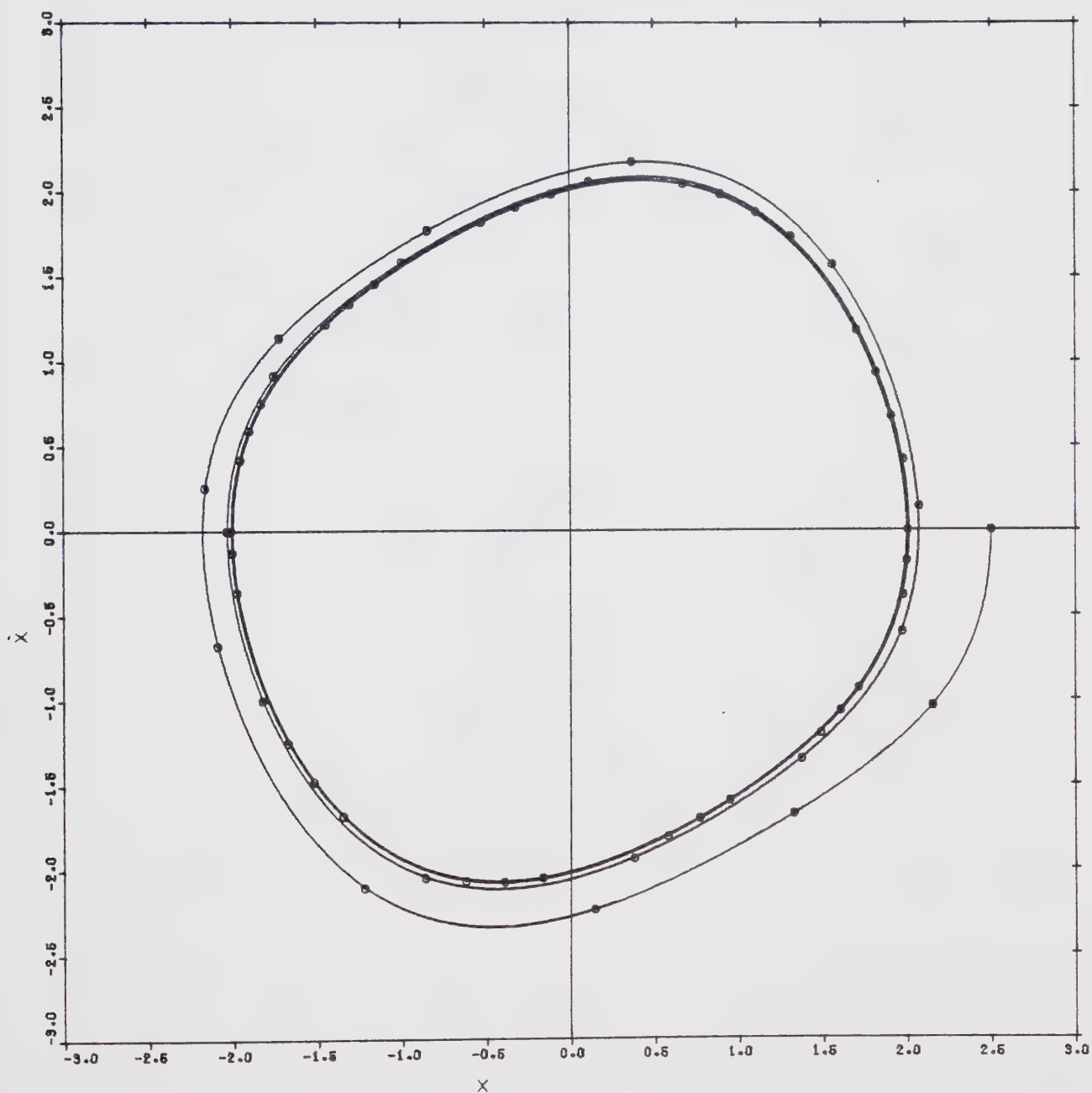
Δt (PIECEWISE
LINEARIZATION) = 0.2000

Δt (RUNGE-KUTTA) = 0.0500

$$x(0) = 2.5000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

○ PIECEWISE LINEARIZATION



[39].) The development of the limit cycle can be seen in the phase-plane diagrams and there appear to be no major difficulties in achieving the desired results. As expected for an initial displacement outside the limit cycle, the solution moves in towards it, while moving out to it when X_0 is inside it. Even for the initial displacement being on the limit cycle, there seem to be no problems.

The next thing was to increase the value of μ while keeping the values of the respective time intervals the same. It had also been decided to restrict the remaining investigation to the situation of the initial displacement outside the limit cycle, except for μ of 5.0 where X_0 will be varied as had been done for μ of 0.25.

The results of this can be seen in Figures 5.9 - 5.18 [38], [39], but something unusual appears in Figures 5.13 - 5.18 [38], [39]. Up to this point, the plots had given good fits to the solution being compared (with some minor deviations for μ of 2.0), while for these curves, this does not seem to be the case. Immediately the piecewise linearization was suspected to be at fault and both the derivation and the computer program were checked several times. Nothing that could introduce what would appear to be a phase difference was found. Internal roundoff error in the computer was also suspected, and the tolerances were reset to smaller values and, as far as possible, the program was reset to operate in double precision. But, the results were still the same--a phase difference, or what appeared to be as such, somehow arose in the system.

A smaller time interval of 0.05 was tried for the piecewise linearization, with the results for $x(0) = 2.5$ and $\mu = 5.0$ to be seen in Figures 5.19 and 5.20 [38], [39]. This seems to be the key to the

FIGURE 5.9 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 1.0, \Delta t_P = 0.20, \Delta t_{RK} = 0.05, x(0) = 2.5,$
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)

$$\mu = 1.0000$$

Δt (PIECEWISE
LINEARIZATION) = 0.2000

Δt (RUNGE-KUTTA) = 0.0500

$$x(0) = 2.5000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

○ PIECEWISE LINEARIZATION

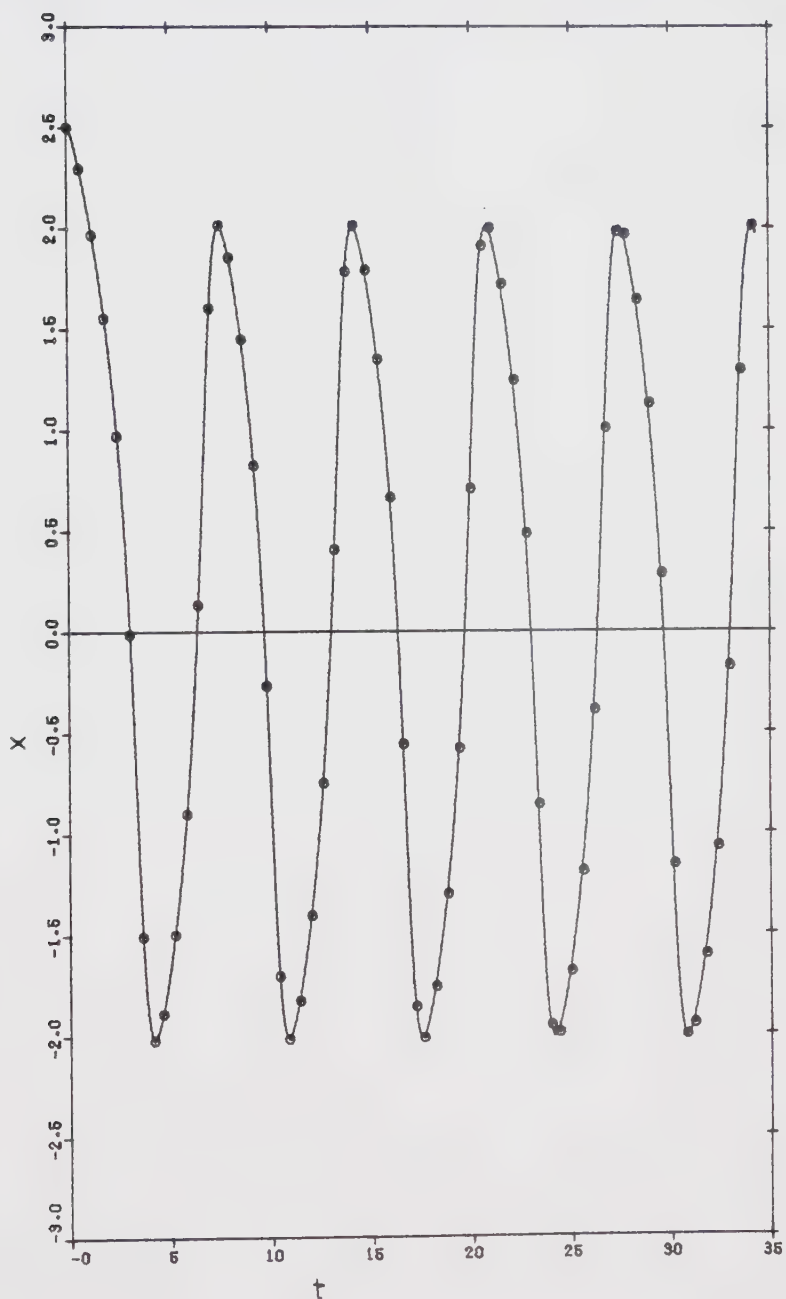


FIGURE 5.10 - PHASE TRAJECTORIES FOR VAN DER POL
EQUATION FOR $\mu = 1.0$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$,
 $x(0) = 2.5$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [39].)

$$\mu = 1.0000$$

Δt (PIECEWISE
LINEARIZATION) = 0.2000

Δt (RUNGE-KUTTA) = 0.0500

$$x(0) = 2.5000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

○ PIECEWISE LINEARIZATION

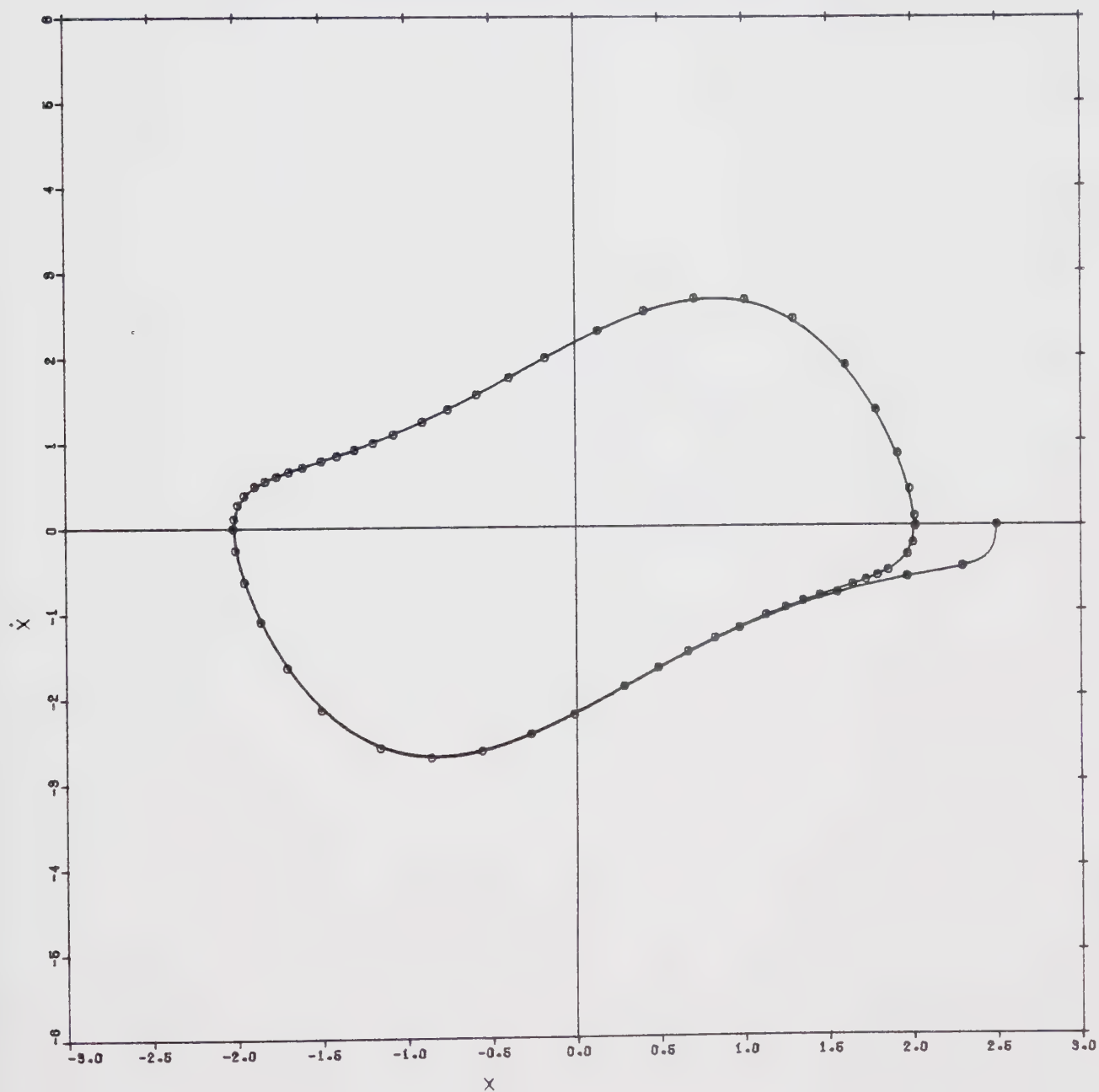


FIGURE 5.11 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 2.0$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$,
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)

$\mu = 2.0000$

Δt (PIECEWISE
LINEARIZATION) = 0.2000

Δt (RUNGE-KUTTA) = 0.0500

$x(0) = 2.5000 \quad \dot{x}(0) = 0.0000$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION

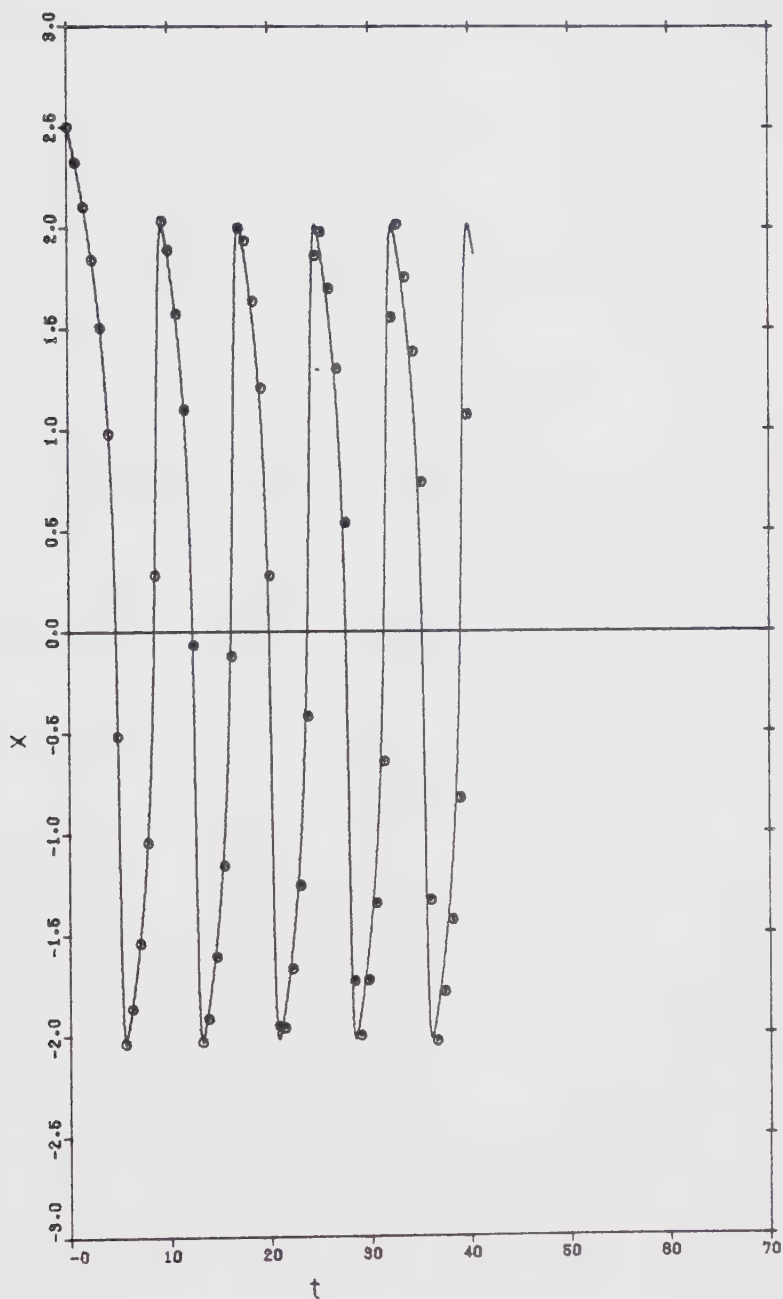


FIGURE 5.12 - PHASE TRAJECTORIES FOR VAN DER POL
EQUATION FOR $\mu = 2.0$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$,
 $x(0) = 2.5$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [39].)

$$\mu = 2.0000$$

Δt (PIECEWISE
LINEARIZATION) = 0.2000

Δt (RUNGE-KUTTA) = 0.0500

$$x(0) = 2.5000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

○ PIECEWISE LINEARIZATION

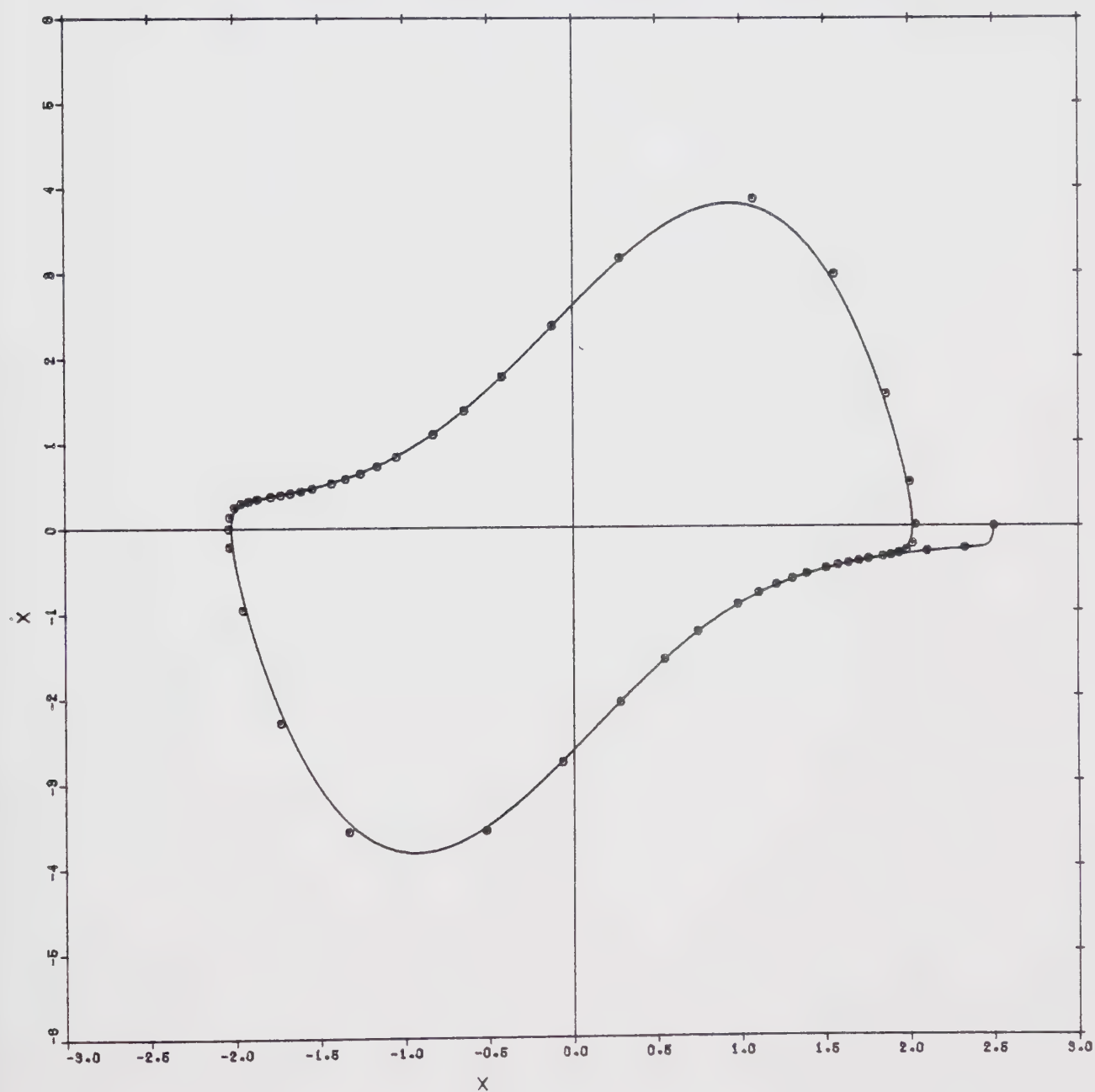


FIGURE 5.13 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 5.0$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 1.0$,
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)

$$\mu = 5.0000$$

Δt (PIECEWISE
LINEARIZATION) = 0.2000

Δt (RUNGE-KUTTA) = 0.0500

$$x(0) = 1.0000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

⊙ PIECEWISE LINEARIZATION

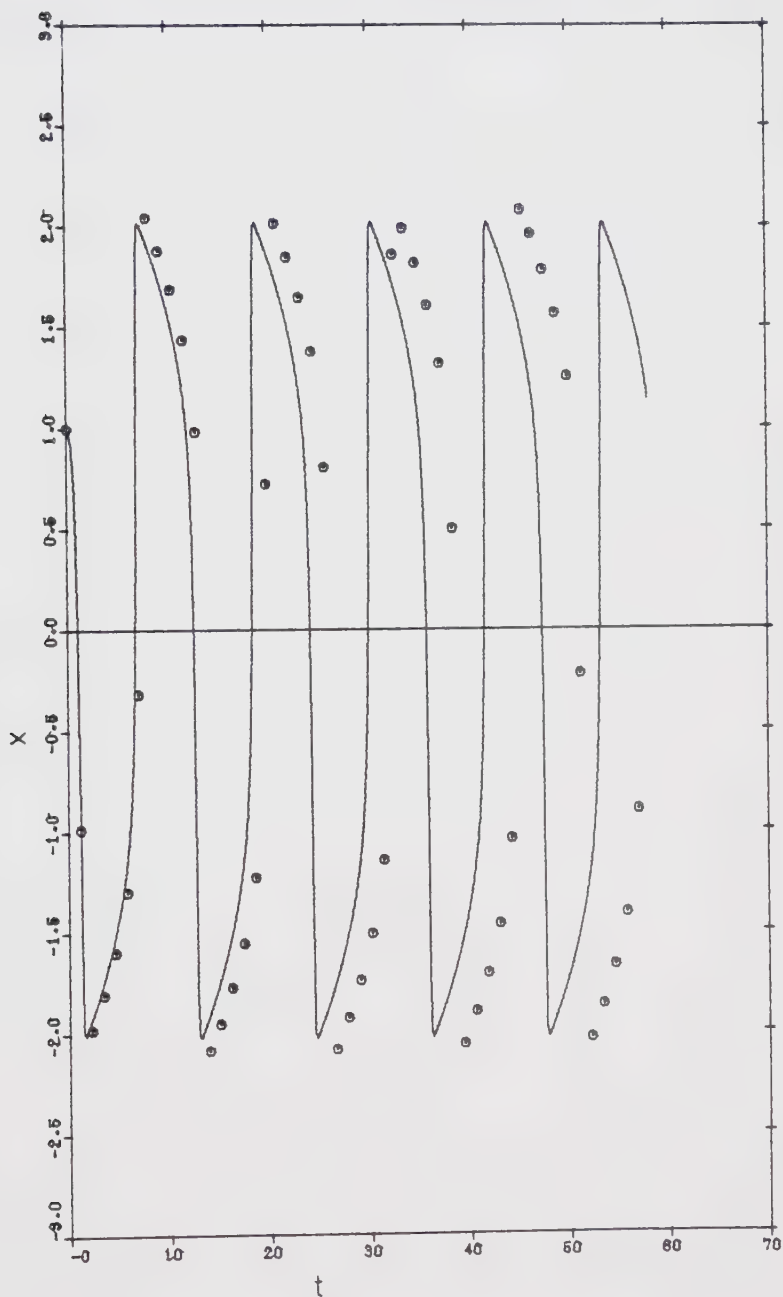


FIGURE 5.14 - PHASE TRAJECTORIES FOR VAN DER POL
EQUATION FOR $\mu = 5.0$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$,
 $x(0) = 1.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [39].)

$$\mu = 5.0000$$

Δt (PIECEWISE
LINEARIZATION) = 0.2000

Δt (RUNGE-KUTTA) = 0.0500

$$x(0) = 1.0000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

⊙ PIECEWISE LINEARIZATION

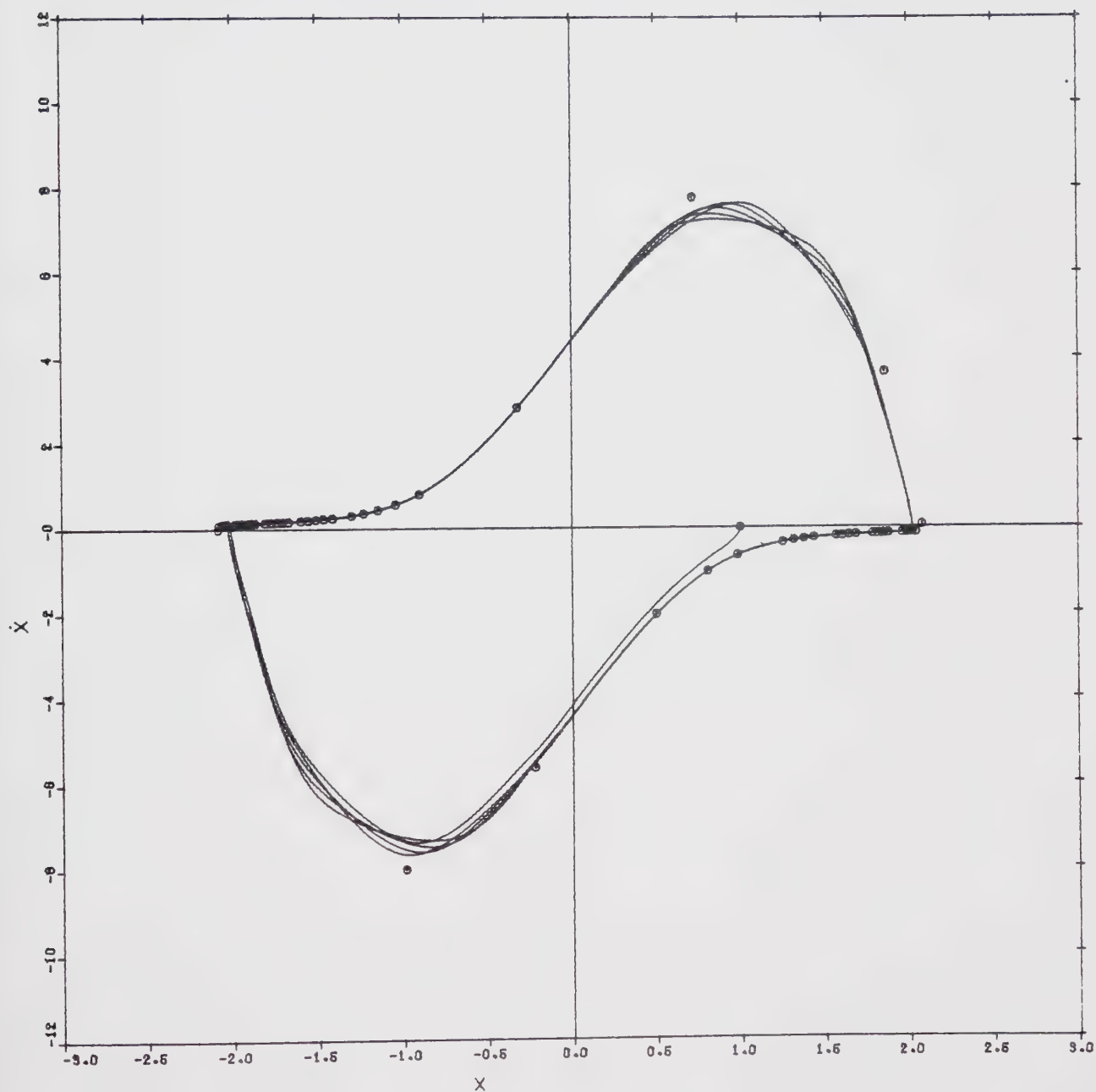


FIGURE 5.15 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 5.0$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$, $x(0) = 2.0$,
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)

$$\mu = 5.0000$$

Δt (PIECEWISE
LINEARIZATION) = 0.2000

Δt (RUNGE-KUTTA) = 0.0500

$$x(0) = 2.0000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

⊙ PIECEWISE LINEARIZATION

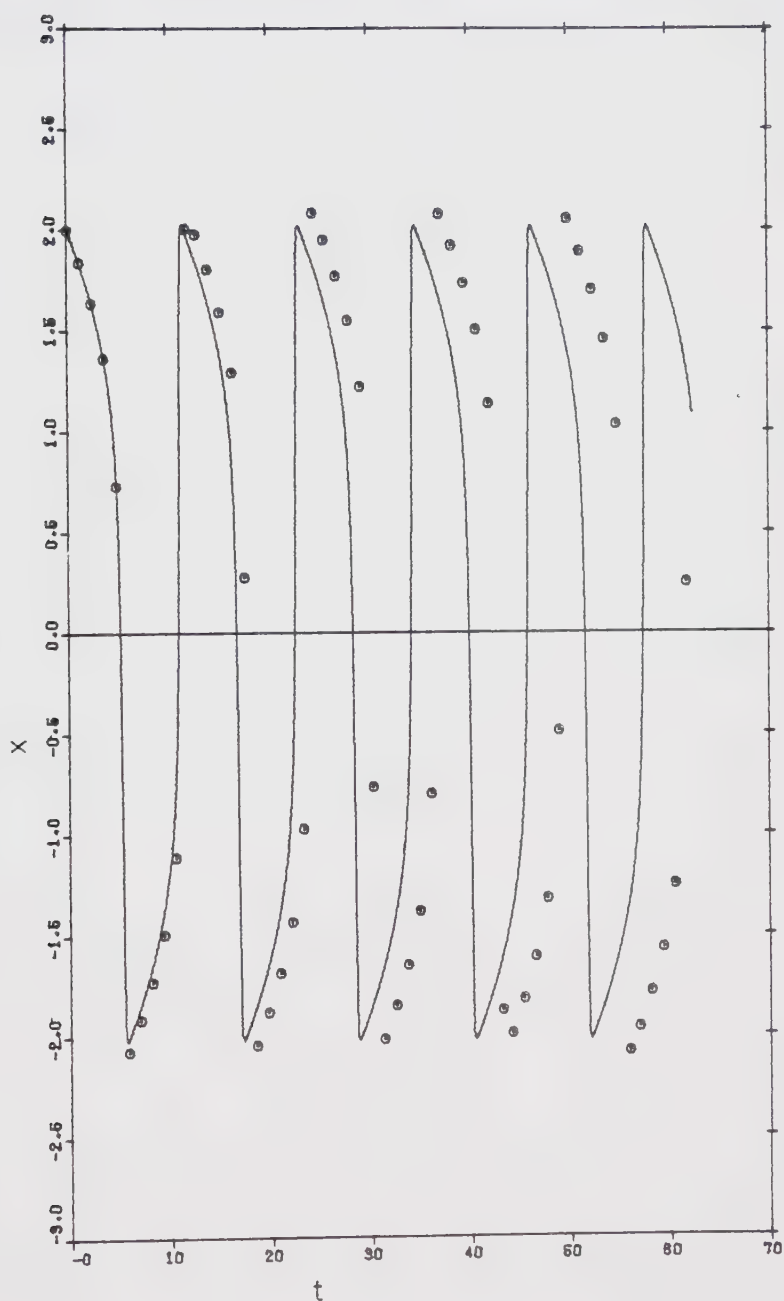


FIGURE 5.16 - PHASE TRAJECTORIES FOR VAN DER POL
EQUATION FOR $\mu = 5.0$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$,
 $x(0) = 2.0$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [39].)

$$\mu = 5.0000$$

Δt (PIECEWISE
LINEARIZATION) = 0.2000

Δt (RUNGE-KUTTA) = 0.0500

$$x(0) = 2.0000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

○ PIECEWISE LINEARIZATION

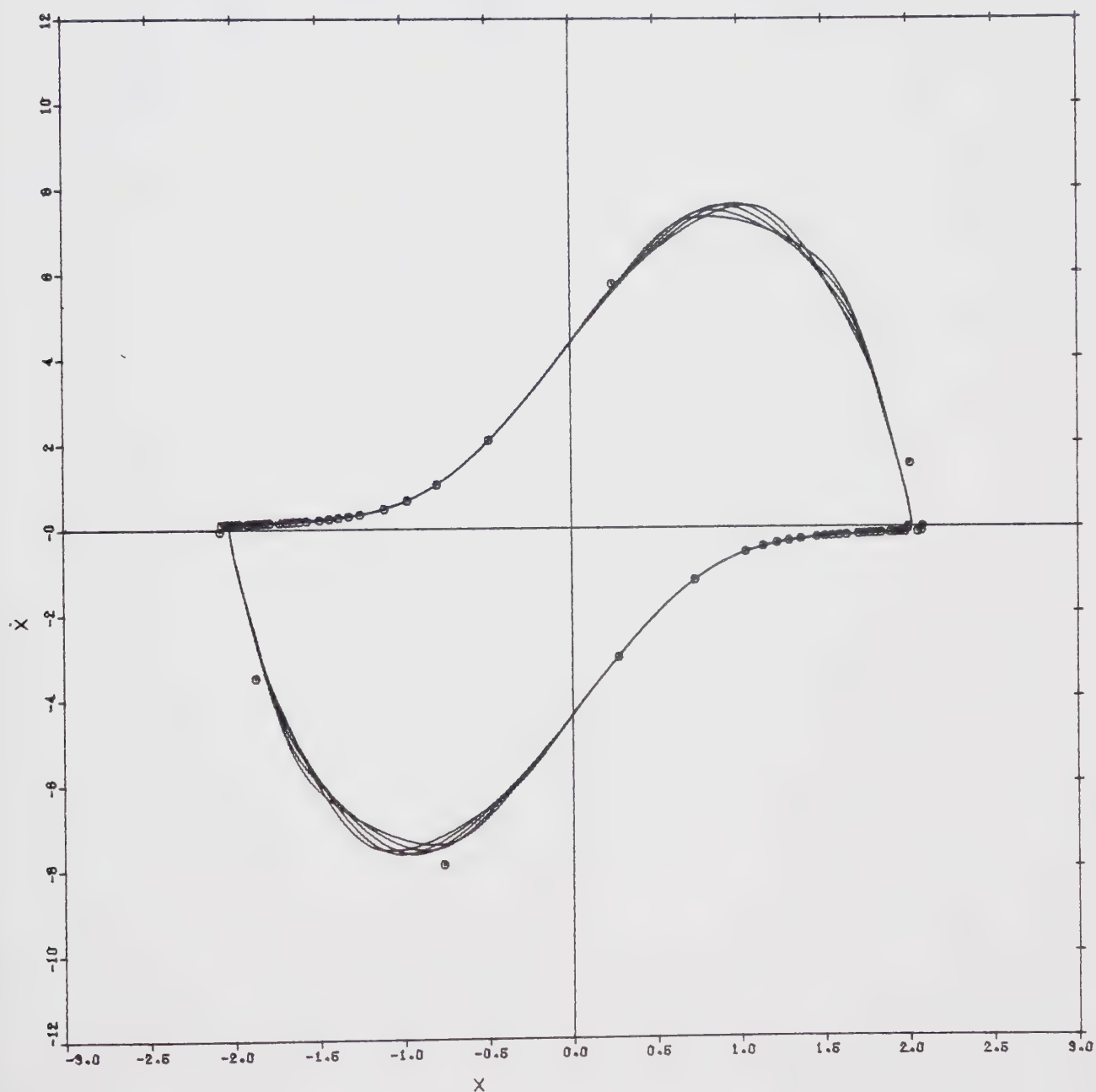


FIGURE 5.17 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 5.0, \Delta t_P = 0.20, \Delta t_{RK} = 0.05, x(0) = 2.5,$
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)

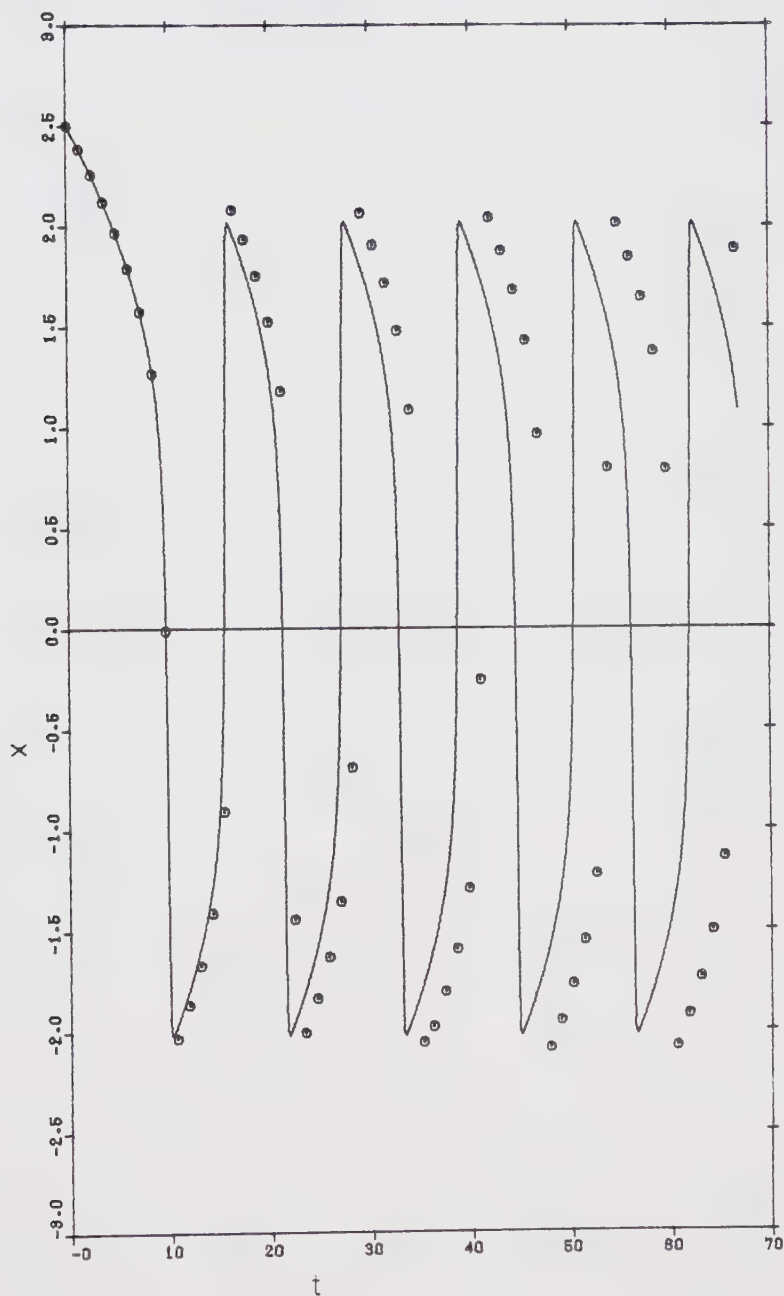
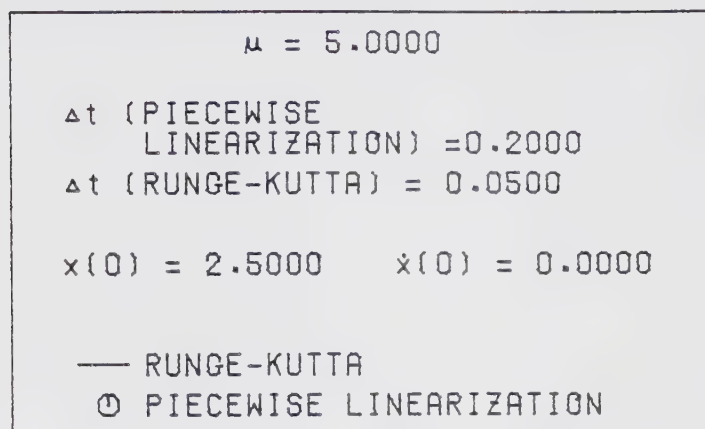


FIGURE 5.18 - PHASE TRAJECTORIES FOR VAN DER POL
EQUATION FOR $\mu = 5.0$, $\Delta t_P = 0.20$, $\Delta t_{RK} = 0.05$,
 $x(0) = 2.5$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [39].)

$$\mu = 5.0000$$

Δt (PIECEWISE
LINEARIZATION) = 0.2000

Δt (RUNGE-KUTTA) = 0.0500

$$x(0) = 2.5000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

○ PIECEWISE LINEARIZATION

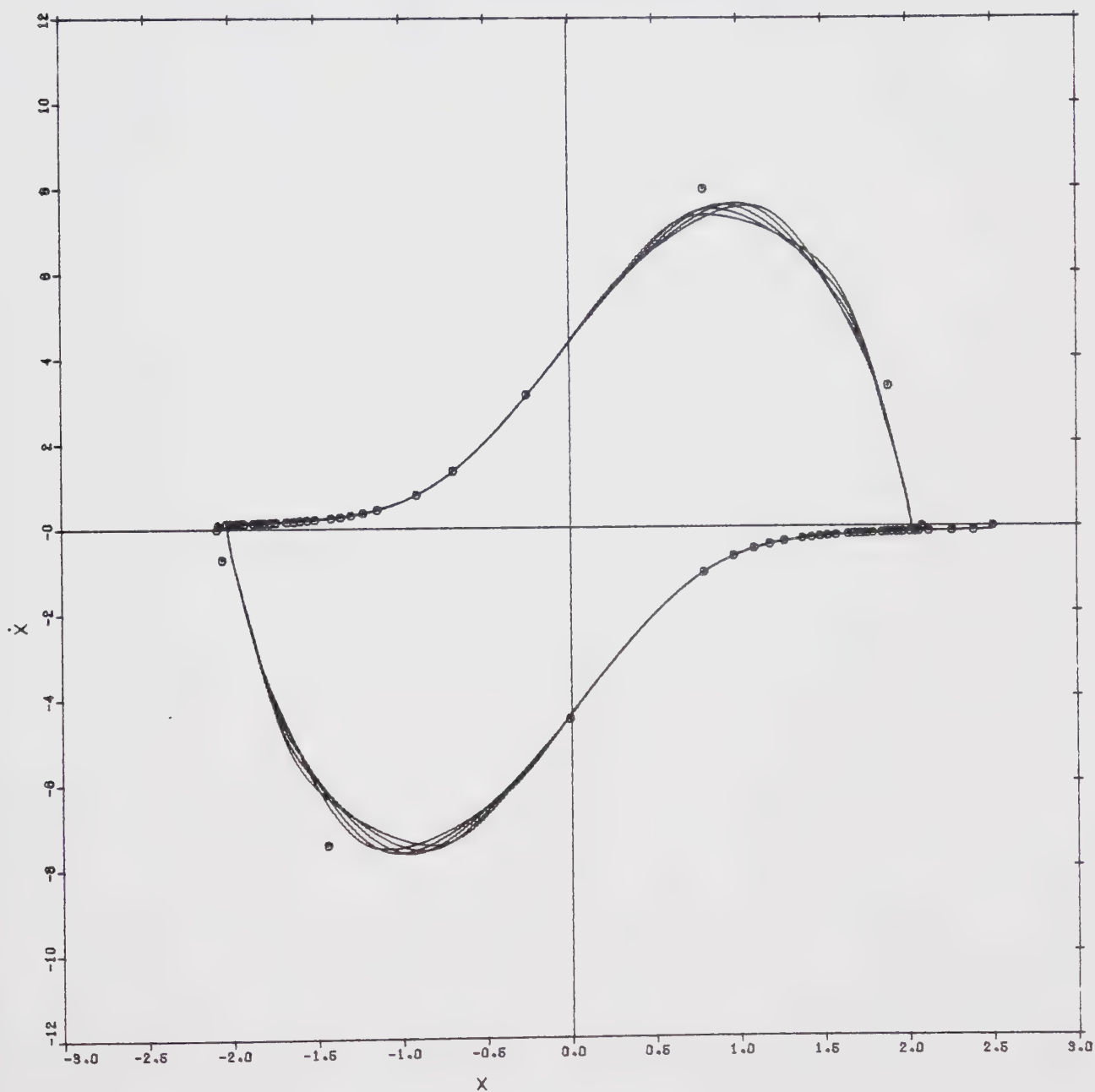


FIGURE 5.19 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 5.0$, $\Delta t_P = 0.05$, $\Delta t_{RK} = 0.05$, $x(0) = 2.5$,
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)

$\mu = 5.0000$

Δt (PIECEWISE
LINEARIZATION) = 0.0500

Δt (RUNGE-KUTTA) = 0.0500

$x(0) = 2.5000 \quad \dot{x}(0) = 0.0000$

— RUNGE-KUTTA
○ PIECEWISE LINEARIZATION

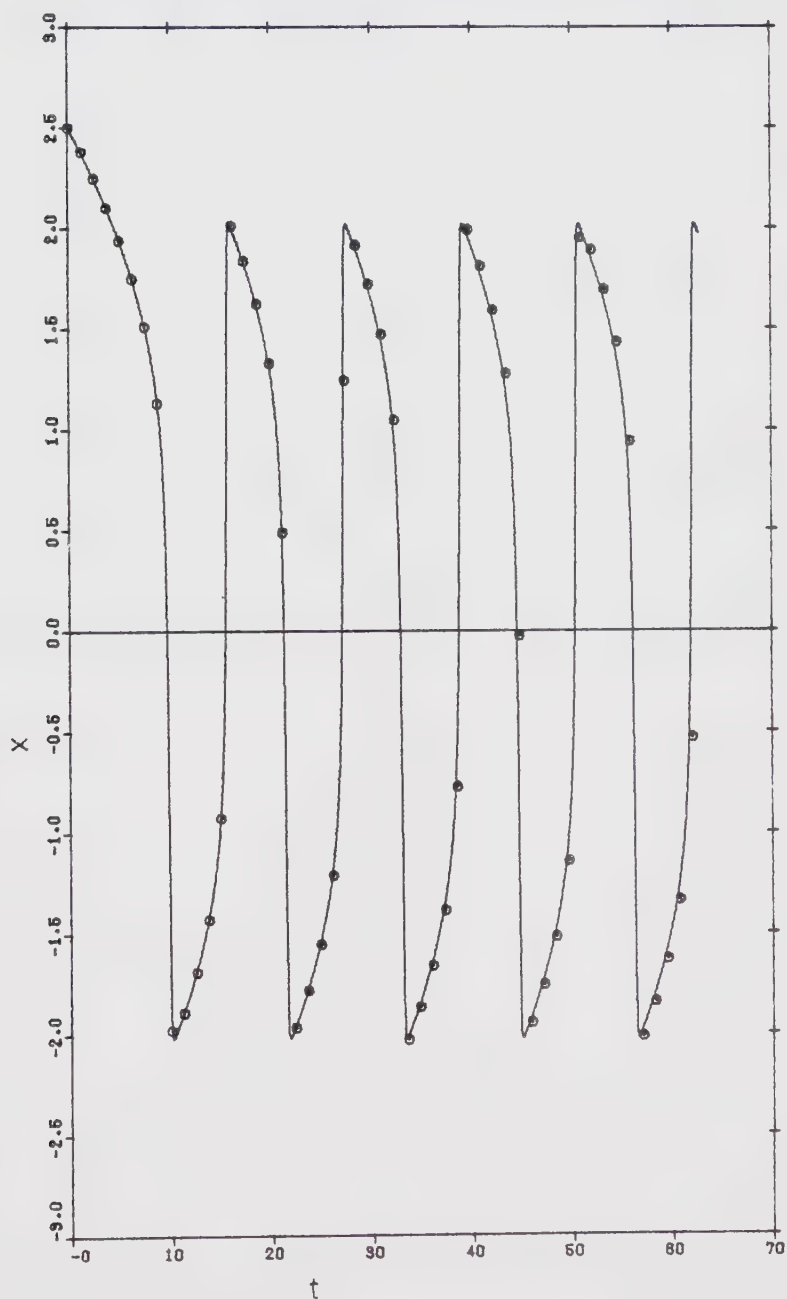


FIGURE 5.20 - PHASE TRAJECTORIES FOR VAN DER POL
EQUATION FOR $\mu = 5.0$, $\Delta t_P = 0.05$, $\Delta t_{RK} = 0.05$,
 $x(0) = 2.5$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [39].)

$$\mu = 5.0000$$

Δt (PIECEWISE
LINEARIZATION) = 0.0500

Δt (RUNGE-KUTTA) = 0.0500

$$x(0) = 2.5000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

○ PIECEWISE LINEARIZATION

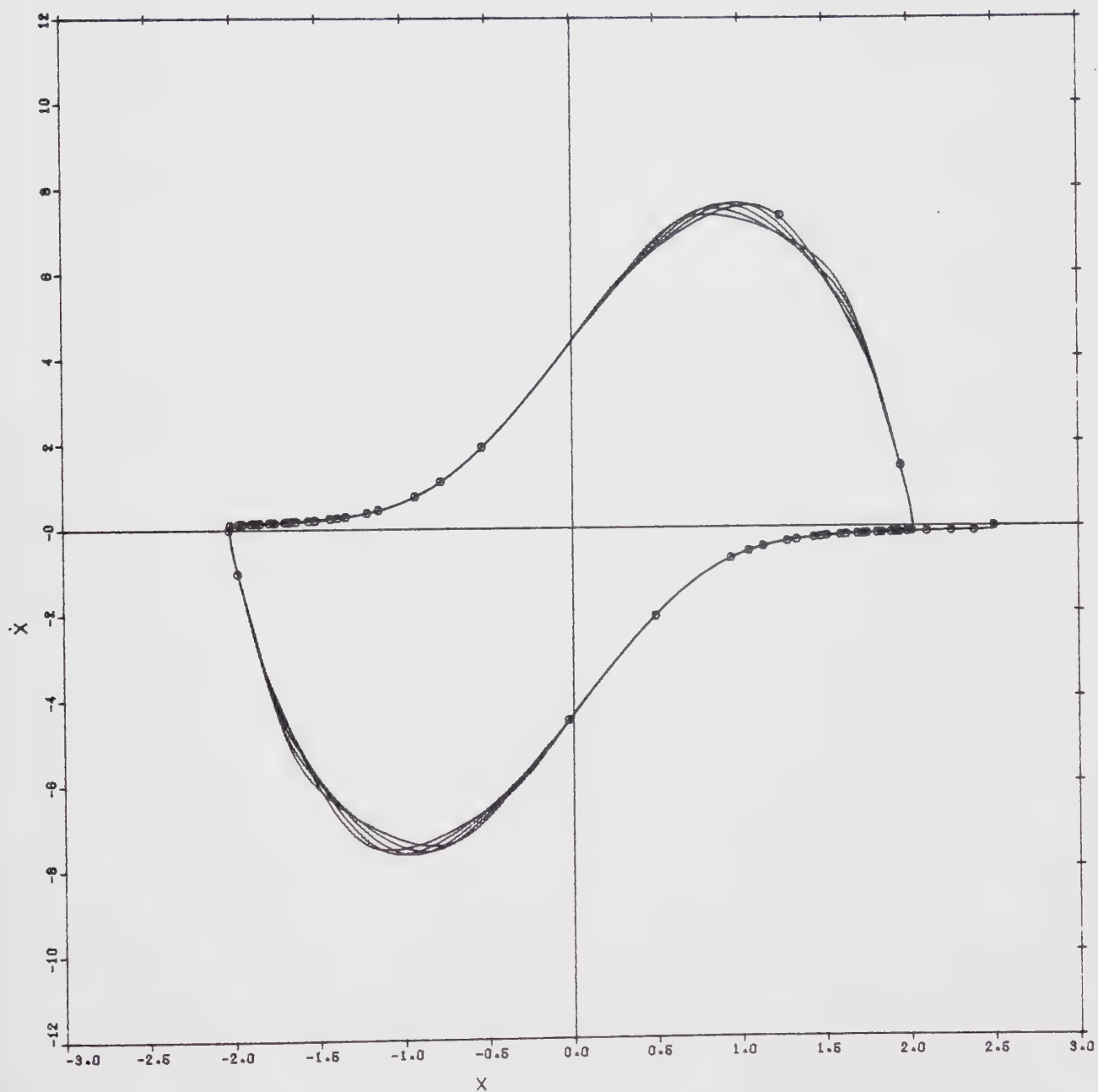


FIGURE 5.21 - SOLUTIONS TO VAN DER POL EQUATION FOR
 $\mu = 5.0, \Delta t_P = 0.25, \Delta t_{RK} = 0.25, x(0) = 2.5,$
 $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [38].)

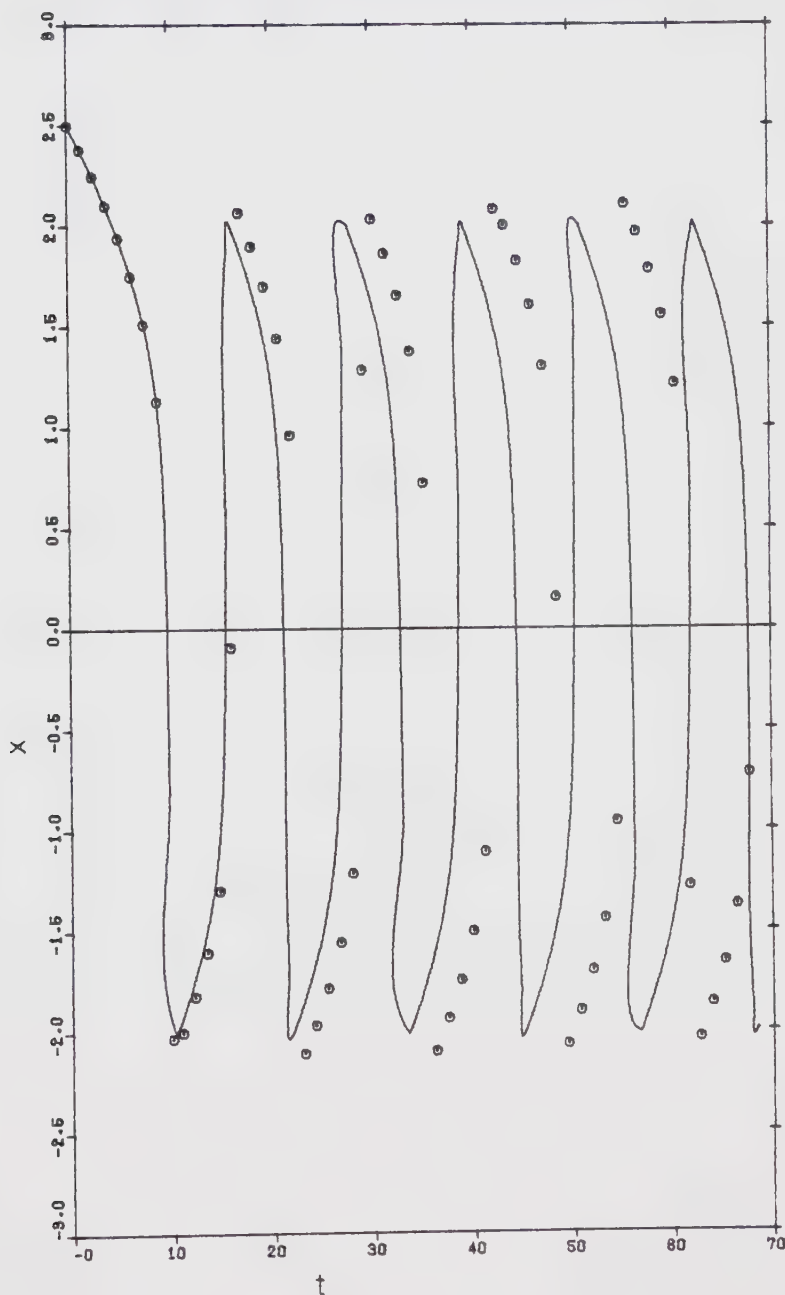
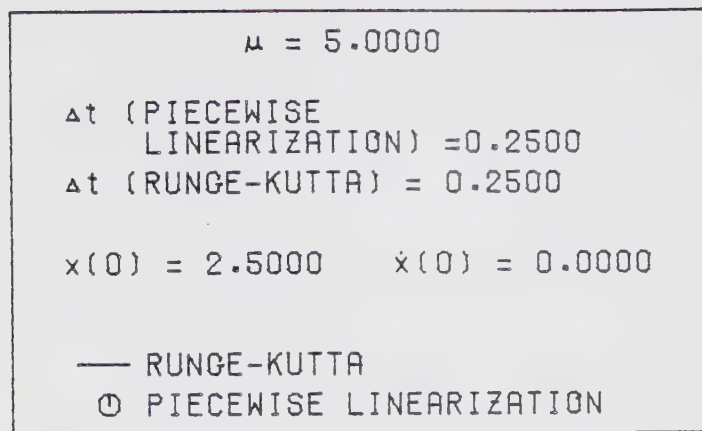


FIGURE 5.22 - PHASE TRAJECTORIES FOR VAN DER POL
EQUATION FOR $\mu = 5.0$, $\Delta t_P = 0.25$, $\Delta t_{RK} = 0.25$,
 $x(0) = 2.5$, $\dot{x}(0) = 0.0$

(N.B.: Runge-Kutta solution uses [5]. Diagram
based on [39].)

$$\mu = 5.0000$$

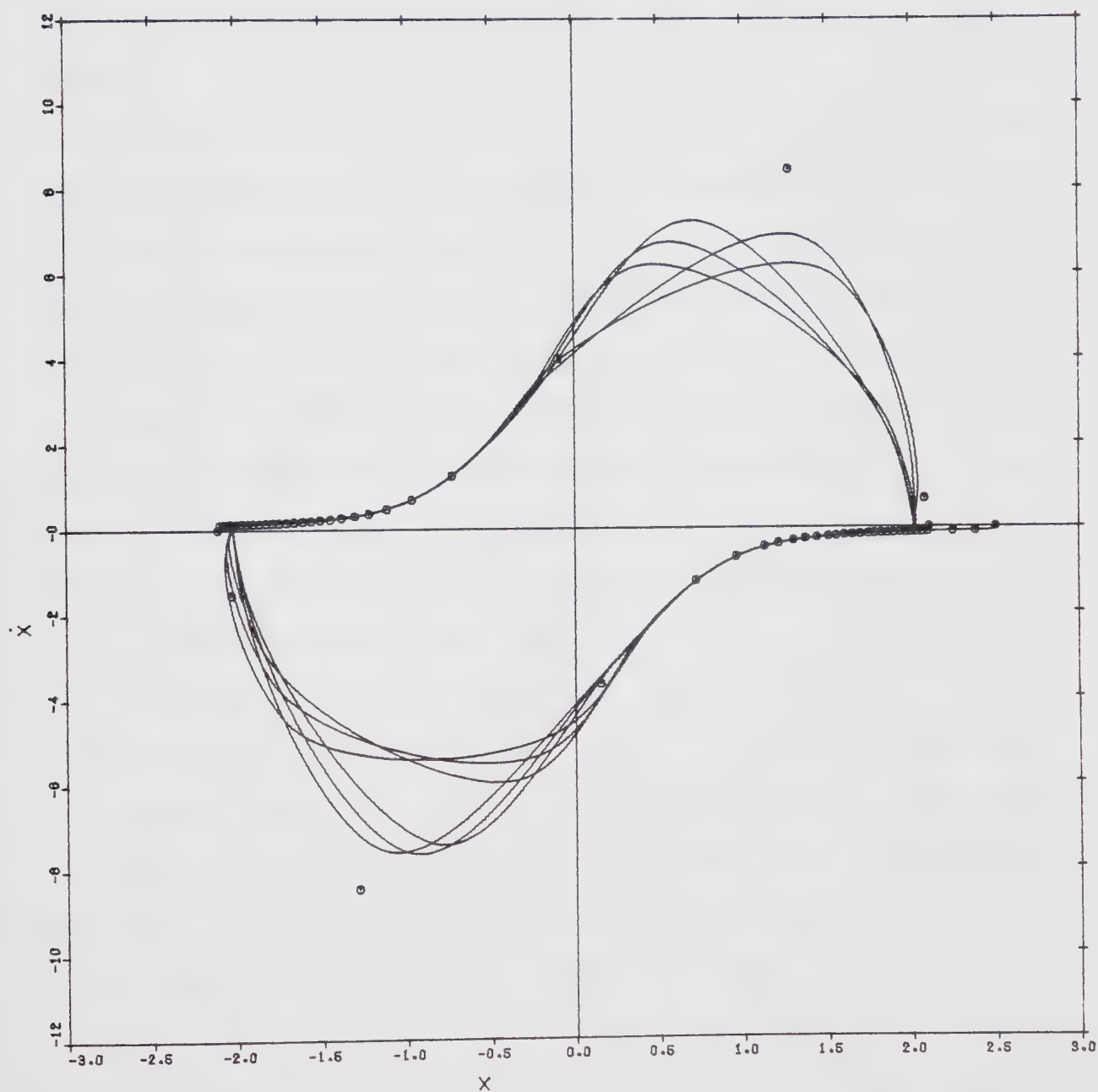
Δt (PIECEWISE
LINEARIZATION) = 0.2500

Δt (RUNGE-KUTTA) = 0.2500

$$\dot{x}(0) = 2.5000 \quad \dot{x}(0) = 0.0000$$

— RUNGE-KUTTA

○ PIECEWISE LINEARIZATION



problem, since the situation appeared to be rectified to some extent in these diagrams, and completely resolved for $\mu = 2.0$ over the time span examined though the plots in question are not shown here.

This raises some interesting questions concerning the solution of this equation. One is concerned with stability, but this will be examined later as a similar problem occurs in the next chapter. Another is that if the piecewise linear method is affected by this behaviour, then the Runge-Kutta solution [5] as used here may, at times, also be suspect.

An example of the latter can be seen in Figures 5.21 and 5.22 [38], [39]. The time interval for each solution is rather large (keeping μ at 5.0), and some distortion can be seen in the displacement-time curve, while the phase plane diagram is very much affected by this to the extent that one cannot tell which part of the curve is the limit cycle.

Figure 5.21 [38] displays a paradox, as there would appear to be more than one value for the displacement as calculated by DVERK [5] for a given time (such as for t of about 23, where the value of x would be about 2.1, 1.6, and -0.9). This is physically impossible, and can lead to an erroneous conclusion about the results.

A possible source may be that the graphics routine used [40] was unable to produce a smooth curve due to a low number of data points being plotted since Δt was large. (By comparison, the curve for Figure 5.19 [38] was more accurate since a much larger number of values had been used.) The examination of the calculated numerical values may give further insight as to what may have brought this about.

Of interest now are the execution times for the two solutions. As before, the program was stripped of all non-essential steps, such as

comment statements and write commands and then run for various values of μ , Δt_p , and Δt_{RK} , with $x(0) = 3.0$ and $\dot{x}(0) = 0.0$ using [27]. The runs were for the time taken to complete 10 half-cycles for the piecewise linearization and the closest time to that value for the Runge-Kutta solution [5]. These were obtained by running an earlier version of the program on the Amdahl 470V/7 [25], with the results seen in Figures 5.23 - 5.26. As expected for this system, the Runge-Kutta [5] results were significantly lower than those for the piecewise linearization, but noteworthy is the fact that the times go up for the piecewise linear method after $\Delta t_p = 0.25$ for $\mu = 2.0$ and $\mu = 5.0$. The calculations for the damping coefficient described in the derivation may explain why this arises, because of the number of iterations required to achieve a solution. Also, for these values of μ , no results were to be obtained for $\Delta t_p = 0.5$, as the program kept exceeding its set time limit, even after a setting of 15.0 seconds [31]. It was decided to abandon any further attempts as the runs would have proved too costly, but this might indicate that an economic upper limit may exist for these values of μ .

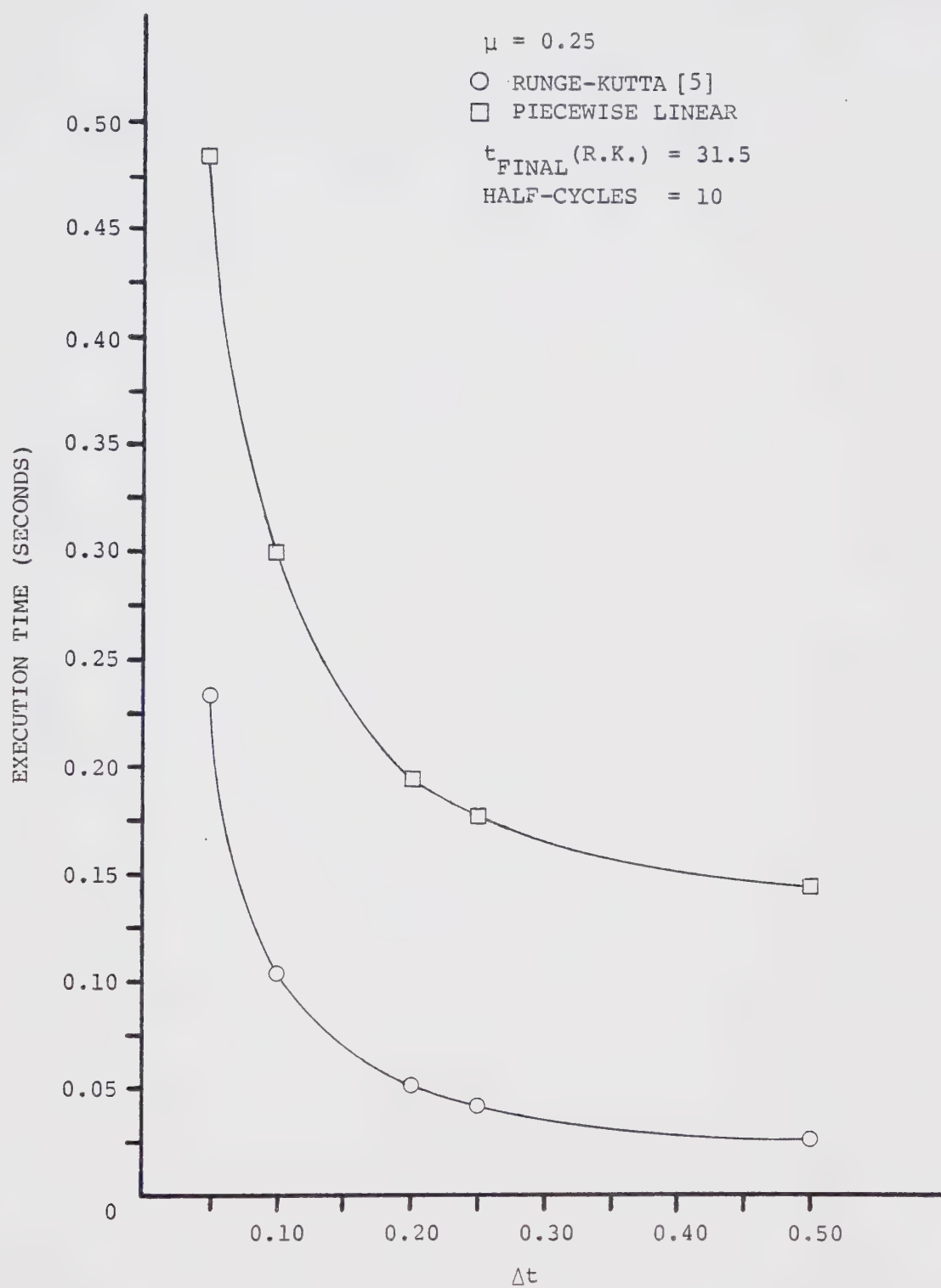


FIGURE 5.23 - EXECUTION TIMES FOR VAN DER POL
 EQUATION SOLUTIONS FOR $\mu = 0.25$, $x(0) = 3.0$,
 $\dot{x}(0) = 0.0$ [27]

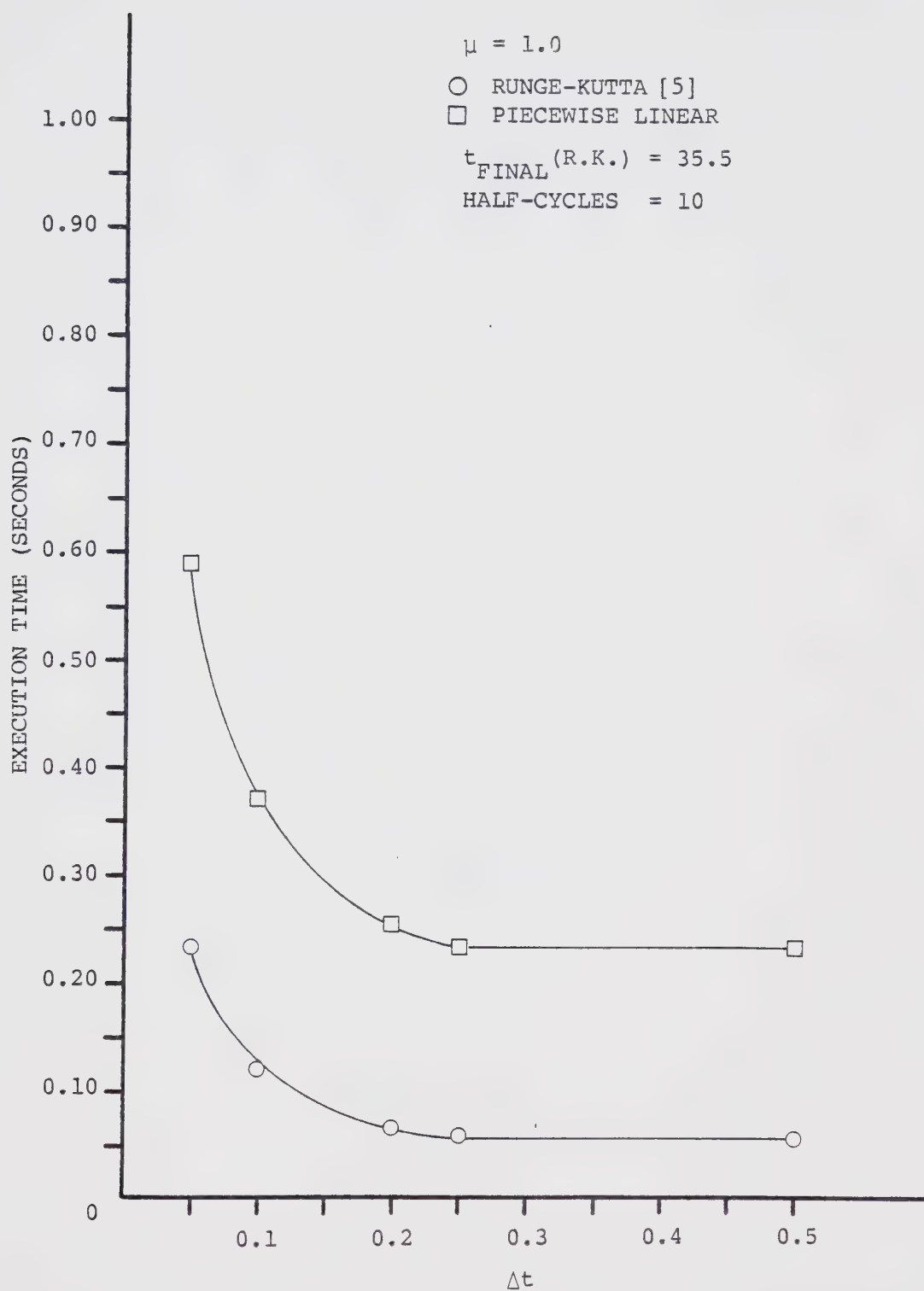


FIGURE 5.24 - EXECUTION TIMES FOR VAN DER POL
 EQUATION SOLUTIONS FOR $\mu = 1.0$, $x(0) = 3.0$,
 $\dot{x}(0) = 0.0$, [27]

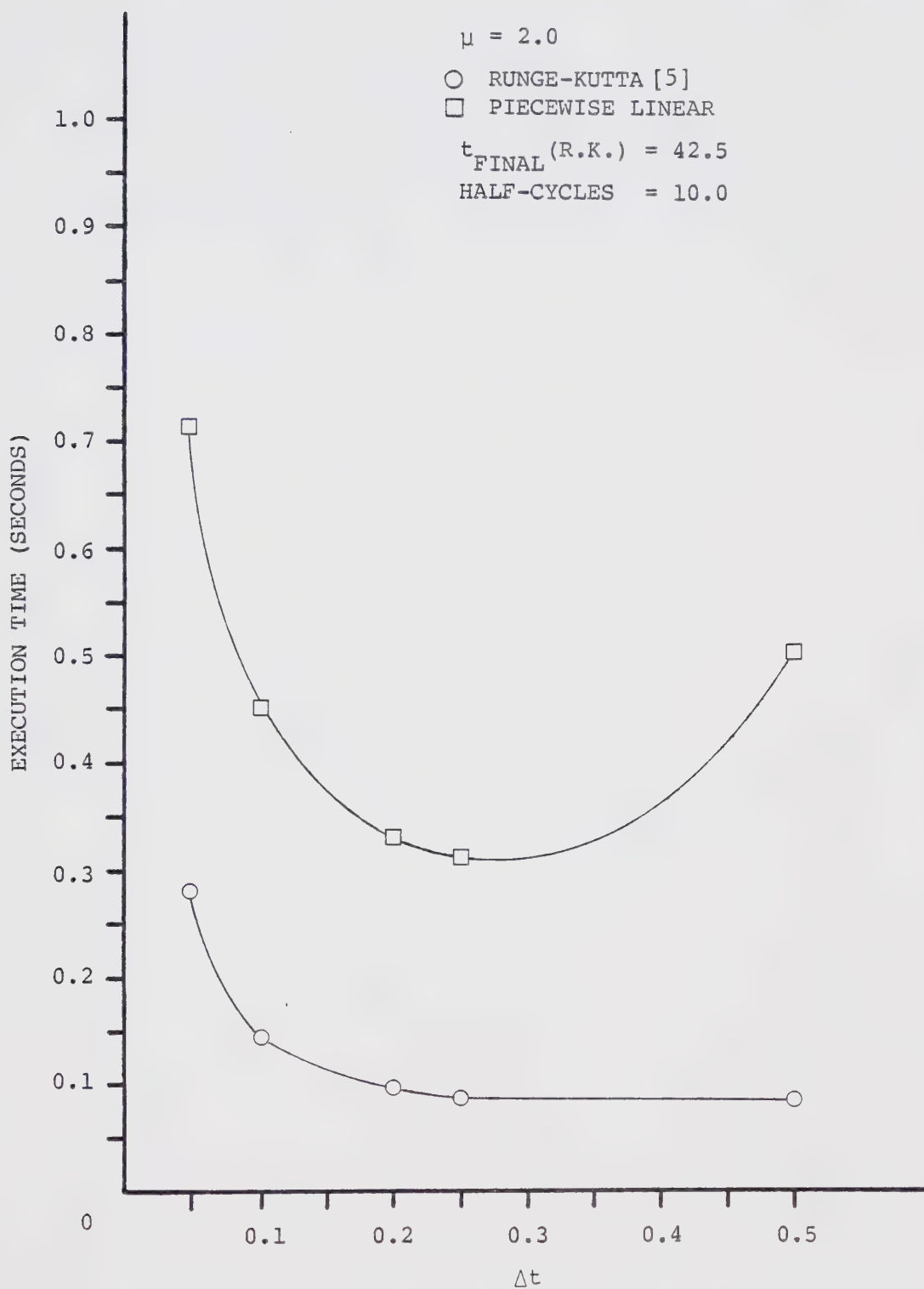


FIGURE 5.25 - EXECUTION TIMES FOR VAN DER POL
 EQUATION SOLUTIONS FOR $\mu = 2.0$, $x(0) = 3.0$,
 $\dot{x}(0) = 0.0$ [27]

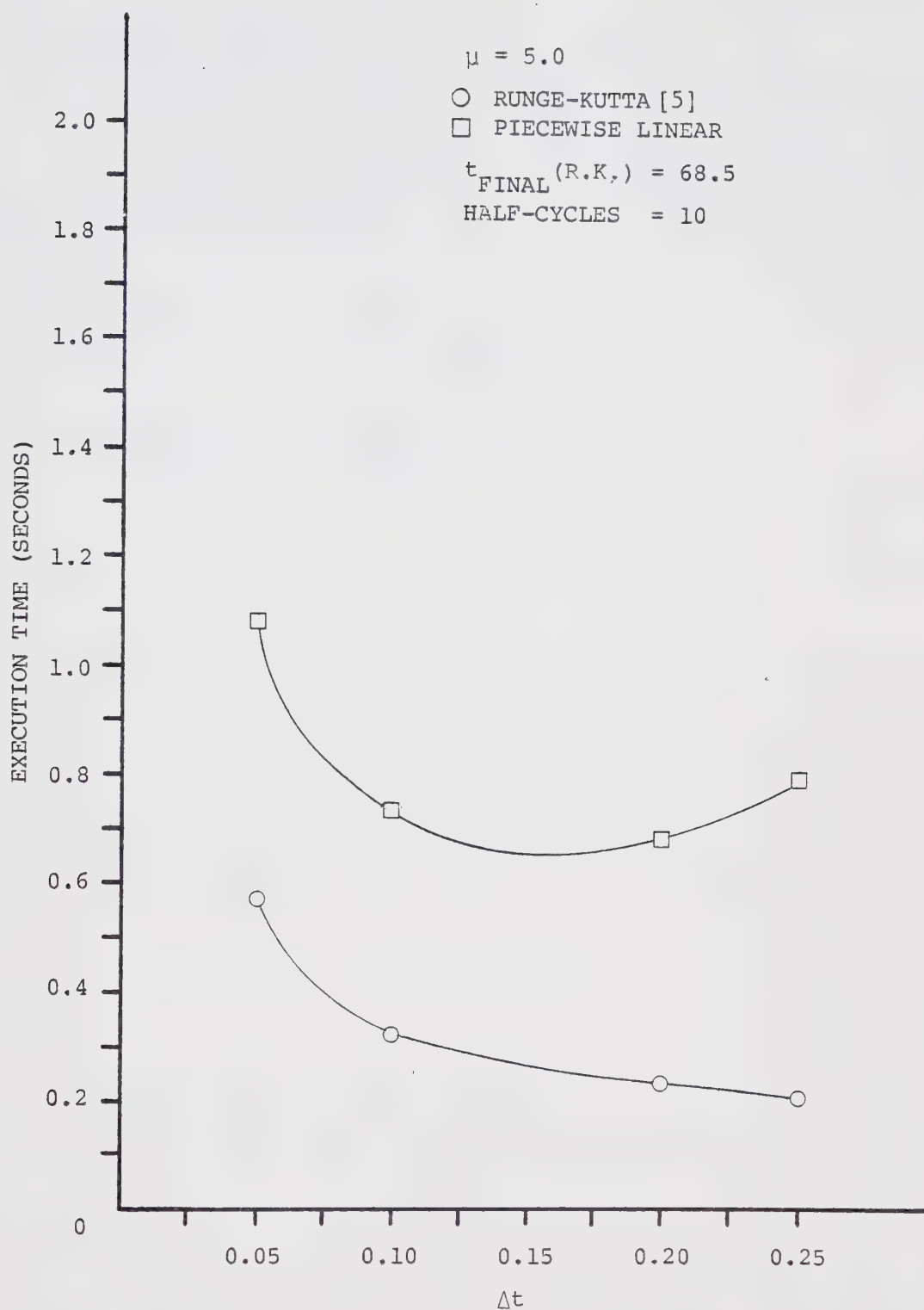


FIGURE 5.26 - EXECUTION TIMES FOR VAN DER POL
 EQUATION SOLUTIONS FOR $\mu = 5.0$, $x(0) = 3.0$,
 $\dot{x}(0) = 0.0$ [27]

VI. MATHIEU EQUATION

A. Preliminary Comments

In this chapter, an example of a linear equation with a periodically varying coefficient, namely the Mathieu equation, is examined. It has the canonical form [41]:

$$y'' + (a - 2q \cos 2z)y = 0 , \quad (6.1.1)$$

a = Characteristic number.

The initial conditions considered are:

$$y(0) = y_0 , \quad y'(0) = y'_0 . \quad (6.1.2)$$

Fortunately, an exact solution exists in terms of some special functions, and a means of generating a series approximation to them will be derived. This will be used as the basis by which to judge the piecewise linearization. The same Runge-Kutta routine [5] that had been used in the earlier chapters will also be used as confirmation of the results. The investigation will be limited to those cases of integral order.

B. Generation of Characteristic Number

The major references for this chapter are Ince [42] and McLachlan [43].

The appropriate characteristic number depends upon the initial conditions from which the solution is generated, the order of the solution (which determines the period τ) and is summarized as [44]:

$$\begin{array}{llll}
 \tau = \pi & & \tau = 2\pi & \\
 y(0) = y_0, & y'(0) = 0 & a_{2n} & a_{2n+1} \\
 y(0) = 0, & y'(0) = y'_0 & b_{2n+2} & b_{2n+1}
 \end{array}$$

One method for this makes use of the recurrence relations for the coefficients of the series form of the exact solutions [45] which are shown below.

For a_{2n} :

$$ce_{2n}(z, q) = \sum_{r=0}^{\infty} A_{2r}^{(2n)} \cos(2rz) . \quad (6.2.1)$$

For a_{2n+1} :

$$ce_{2n+1}(z, q) = \sum_{r=0}^{\infty} A_{2r+1}^{(2n+1)} \cos(2r+1)z . \quad (6.2.2)$$

For b_{2n+1} :

$$se_{2n+1}(z, q) = \sum_{r=0}^{\infty} B_{2r+1}^{(2n+1)} \sin(2r+1)z . \quad (6.2.3)$$

For b_{2n+2} :

$$se_{2n+2}(z, q) = \sum_{r=0}^{\infty} B_{2r+2}^{(2n+2)} \sin(2r+2)z . \quad (6.2.4)$$

The coefficients for the equations are calculated using the recurrence relations described in McLachlan [46].

For $ce_{2n}(z, q)$:

$$\begin{aligned}
 aA_0 - qA_2 &= 0, \\
 (a - 4)A_2 - q(A_4 + 2A_0) &= 0, \\
 (a - 4r^2)A_{2r} - q(A_{2r+2} + A_{2r-2}) &= 0, \\
 (r \geq 2).
 \end{aligned}
 \quad \left. \vphantom{\begin{aligned} aA_0 - qA_2 &= 0, \\ (a - 4)A_2 - q(A_4 + 2A_0) &= 0, \\ (a - 4r^2)A_{2r} - q(A_{2r+2} + A_{2r-2}) &= 0, \end{aligned}} \right\} (6.2.5)$$

For $ce_{2n+1}(z, q)$:

$$\begin{aligned}
 (a - 1 - q)A_1 - qA_3 &= 0 \\
 [a - (2r + 1)^2]A_{2r+1} \\
 - q(A_{2r+3} + A_{2r-1}) &= 0, \\
 (r \geq 1).
 \end{aligned}
 \quad \left. \vphantom{\begin{aligned} (a - 1 - q)A_1 - qA_3 &= 0 \\ [a - (2r + 1)^2]A_{2r+1} \\ - q(A_{2r+3} + A_{2r-1}) &= 0, \end{aligned}} \right\} (6.2.6)$$

For $se_{2n+1}(z, q)$:

$$\begin{aligned}
 (b - 1 + q)B_1 - qB_3 &= 0, \\
 [b - (2r + 1)^2]B_{2r+1} \\
 - q(B_{2r+3} + B_{2r-1}) &= 0, \\
 (r \geq 1).
 \end{aligned}
 \quad \left. \vphantom{\begin{aligned} (b - 1 + q)B_1 - qB_3 &= 0, \\ [b - (2r + 1)^2]B_{2r+1} \\ - q(B_{2r+3} + B_{2r-1}) &= 0, \end{aligned}} \right\} (6.2.7)$$

For $se_{2n+2}(z, q)$:

$$\begin{aligned}
 (b - 4)B_2 - qB_4 &= 0, \\
 (b - 4r^2)B_{2r} \\
 - q(B_{2r+2} + B_{2r-2}) &= 0, \\
 (r \geq 2).
 \end{aligned}
 \quad \left. \vphantom{\begin{aligned} (b - 4)B_2 - qB_4 &= 0, \\ (b - 4r^2)B_{2r} \\ - q(B_{2r+2} + B_{2r-2}) &= 0, \end{aligned}} \right\} (6.2.8)$$

With Ince's notation [39] for the continued fraction:

$$\prod_{r=1}^{\infty} \frac{\alpha_r}{\beta_r} = \frac{1}{\beta_r} + \frac{\alpha_{r+1}}{\beta_{r+1}} + \frac{\alpha_{r+2}}{\beta_{r+2}} + \dots$$

and the rearrangement of some of the terms, the above relations lead to the following. For $ce_{2n}(z, q)$, the result is [47]:

$$\frac{A_{2r}}{A_{2r-2}} = -\frac{q}{4r^2} \prod_{r=1}^{\infty} \frac{-\frac{q^2}{16v^2(v-1)^2}}{1 - \frac{a}{4v^2}},$$

which yields the continued fraction [48]:

$$a_{2n} = -\frac{q^2/2}{1 - \frac{a}{4}} - \frac{q^2/64}{1 - \frac{a}{16}} - \frac{q^2/576}{1 - \frac{a}{36}} - \dots \quad (6.2.9)$$

For $se_{2n+1}(z, q)$, the continued fraction is [49]:

$$\frac{B_{2r-1}}{B_{2r+1}} = \frac{-q}{(2r+1)^2} \prod_{r=1}^{\infty} \frac{-\frac{q^2}{(4v^2-1)^2}}{1 - \frac{b}{(2v+1)^2}},$$

which yields the continued fraction for the characteristic number [49]:

$$b_{2n+1} = 1 - q - \frac{q^2/9}{1 - \frac{b}{9}} - \frac{q^2/225}{1 - \frac{b}{25}} - \frac{q^2/1225}{1 - \frac{b}{49}} - \dots \quad (6.2.10)$$

The case of $ce_{2n+1}(z, q)$ is similar to the previous function since the continued fraction for the coefficients is the same, but with a change of sign in q , yielding the value for the characteristic number [49]:

$$a_{2n+1}(q) = b_{2n+1}(-q) . \quad (6.2.11)$$

Finally, for $se_{2n+2}(z, q)$ the continued fraction is [49]:

$$\frac{B_{2r}}{B_{2r-2}} = - \frac{q}{4r^2} \frac{\infty}{\phi} \frac{\frac{q^2}{16v^2(v-1)^2}}{1 - \frac{b}{4v^2}} ,$$

and the characteristic number continued fraction is [49]:

$$b_{2n+2} = 4 - \frac{q^2/16}{1 - \frac{b}{16}} - \frac{q^2/576}{1 - \frac{b}{36}} - \frac{q^2/2304}{1 - \frac{b}{64}} - \dots \quad (6.2.12)$$

The continued fractions have terms of the form [48]:

$$\gamma_n = \frac{f(a, q, n)}{\phi_{n+1}} ,$$

$$\phi_{n+1} = f(a, q, n) - \gamma_{n+1} .$$

Calculation is halted when, for the accuracy required, $\phi_{n+1} = 1$. It was decided that 25 terms would be sufficient.

A characteristic number can now be generated using the continued

fractions just given together with a suggested approach given by Ince [48]. He states how someone doing this task using a calculator, paper and pencil could arrive at a final result. The author took this method and first went through the procedure by hand for tabulated values for the characteristic number and flowcharting the decision processes required, and then testing this method on a Hewlett-Packard HP-67 hand calculator, transferring the final version of the logic to a computer program, which was later run on the Amdahl 470V/8 [26]. The results were compared with the tabulated results given by Ince [50] and there was agreement to seven or eight significant figures.

The following method can be considered as consisting of two parts. The first one (Steps 1-3) is a scan through a range of values for the characteristic number, for an estimated value for a (a_{GUESS}) and a given q , until an upper and a lower bound, between which the final value will lie, are found. The second one (Steps 4-6) is an interval halving search for this final value. The particular continued fraction used is dependent upon the initial conditions in effect. If at any time during the scan a a_{IN} is approximately equal to a a_{OUT} , the process stops.

1. A value for a a_{OUT} is found using a a_{IN} .
2. If a a_{IN} is greater than a a_{OUT} , assume a a_{IN} to be high, decrement it and repeat Step 1 until either a a_{IN} is less than a a_{OUT} , or the sign of the present a a_{OUT} is different than that for the previous a a_{OUT} and the present and previous a a_{IN} . The upper bound (a_{UP}) is the next-to-last a a_{IN} and the lower bound (a_{LOW}) the final one.
3. If a a_{IN} is less than a a_{OUT} , assume a a_{IN} to be low, increment it and repeat Step 1 until a a_{IN} is greater than a a_{OUT} . The upper bound is the final a a_{IN} and the lower one is the previous value.

4. The midpoint a_{MID} is calculated from the average of the upper and lower bounds, and an output value a_{MIDOUT} is found. If a_{MID} is nearly equal to a_{MIDOUT} , a_{MID} is the final value for a . If not, continue below.

5. Consider the case of a positive a_{MID} . If a_{MIDOUT} is less than or equal to zero, a_{LOW} is the present a_{MID} , and Step 4 is repeated. If a_{MIDOUT} is greater than zero and a_{MID} is less than or equal to a_{MIDOUT} , the present a_{MID} becomes a_{LOW} and Step 4 is done again. However, should a_{MIDOUT} be greater than zero, and a_{MID} be greater than a_{MIDOUT} , a_{MID} becomes a_{UP} and the process returns to Step 4.

6. The following applies to a_{MID} less than or equal to zero. If a_{MIDOUT} is greater than zero, a_{MID} becomes the new a_{LOW} , and the next move is to Step 4. For a_{MIDOUT} less than or equal to zero and a_{MID} greater than a_{MIDOUT} , the new a_{UP} is the present a_{MID} , and then to Step 4. If a_{MIDOUT} is less than or equal to zero, and a_{MID} is less than or equal to a_{MIDOUT} , a_{LOW} takes on the value of the present a_{MID} and Step 4 is repeated.

This covers the method for generating a value for a characteristic number. Specifically, the particular calculation chosen depends upon the initial conditions and the proximity of the first guess to the final value.

To elaborate, for a non-zero initial displacement and zero initial velocity, a_{2n} and a_{2n+1} would be the appropriate values, with b_{2n+1} and b_{2b+2} being in effect for zero initial displacement and non-zero initial velocity. The program that was written for this purpose calculated the odd order value first (this choice having been arbitrary) and then the even order one was found. Whichever was closest to a_{GUESS} became the

final result. This was then taken to generate the three solutions used.

C. Series Approximation to Exact Solution

As mentioned earlier, Mathieu functions can be approximated by series solutions [45], which were given as (6.2.1) to (6.2.4). The question now is how to generate the coefficients that were given. According to McLachlan [46], [51], the recurrence relations that were given in (6.2.5) to (6.2.8) can be used. However, a more suitable form for these relations must be obtained in order to facilitate calculation of the coefficients.

Consider $ce_{2n}(z, q)$ [52]. The first relation of (6.2.5) gives:

$$\frac{A_2}{A_0} = \frac{a}{q}.$$

The next one yields:

$$\frac{A_4}{A_2} = \frac{a - 4}{q} - \frac{2}{A_2/A_0}.$$

The final one gives:

$$\frac{A_{2r}}{A_{2r-2}} = \frac{q}{a - 4r^2 - q \left[\frac{A_{2r+2}}{A_{2r}} \right]},$$

$$(r \geq 3).$$

It had been decided to terminate calculation after 25 terms, since ratios beyond A_{52}/A_{50} would become so small that they could be neglected

when compared with the other terms [53]. Once the various ratios have been found, the normalization [54]:

$$1 = 2A_0^2 + A_2^2 + A_4^2 + \dots ,$$

$$\frac{1}{A_0^2} = 2 + \left(\frac{A_2}{A_0} \right)^2 + \left(\frac{A_4}{A_0} \right)^2 + \dots$$

is required, and so the values for A_{2r}/A_0 are needed. This is done by using:

$$\frac{A_{2r}}{A_0} = \frac{A_{2r}}{A_{2r-2}} \cdot \frac{A_{2r-2}}{A_0} ,$$

starting with A_2/A_0 and A_4/A_2 , and working sequentially through all the ratios. Once this has been accomplished, using the different values for A_{2r}/A_0 , A_0 and A_{2r} can be found.

With the various A_{2r} 's, a series approximation for $ce_{2n}(z,q)$ can be generated.

Take now $ce_{2n+1}(z,q)$ [55]. The first ratio from (6.2.6) gives:

$$\frac{A_3}{A_1} = \frac{a - 1 - q}{q} .$$

The remaining ratios are of the form:

$$\frac{A_{2r+1}}{A_{2r-1}} = \frac{q}{a - (2r + 1)^2 - q \left[\frac{A_{2r+3}}{A_{2r+1}} \right]} ,$$

($r \geq 2$).

As was the case for the previous function, calculation was halted

after 25 terms because ratios beyond A_{51}/A_{49} can be considered as being negligible when compared with the others. The resulting normalization used is [56]:

$$1 = A_1^2 + A_3^2 + A_5^2 + \dots ,$$

and the series approximation for $ce_{2n+1}(z,q)$ using the values for A_{2r+1} is found using the same method as before.

Consider now $se_{2n+1}(z,q)$. Owing to the similarities between (6.2.6) and (6.2.7), the derivations can be omitted and the results presented [49]:

$$\frac{B_3}{B_1} = \frac{b - 1 + q}{q} ,$$

$$\frac{B_{2r+1}}{B_{2r-1}} = \frac{q}{b - (2r + 1)^2 - q \left[\frac{B_{2r+3}}{B_{2r-1}} \right]} ,$$

$$(r \geq 2) .$$

$$1 = B_1^2 + B_3^2 + B_5^2 + \dots .$$

From this, $se_{2n+1}(z,q)$ can be found since the coefficients can be determined from the above relations.

Finally, the results for $se_{2n+2}(z,q)$ will be shown. Since (6.2.8) is similar to (6.2.5), the ratios are as follows, since the derivations would be alike [55], [56]:

$$\frac{B_4}{B_2} = \frac{b - 4}{q} ,$$

$$\frac{B_{2r}}{B_{2r-2}} = \frac{q}{b - 4r^2 - q \left[\frac{B_{2r+2}}{B_{2r}} \right]},$$

$$(r \geq 3).$$

$$1 = B_2^2 + B_4^2 + B_6^2 + \dots$$

From this, an approximation to a Mathieu function of integral order can be generated, thus allowing one to consider the complete solution to the original equation. A computer program was written to make use of the relations just derived, and the results were checked against tabulated values that Ince presented [57], and were found to be in agreement to at least seven significant figures in almost all cases.

Using these coefficients, (6.2.1) to (6.2.4) can now be calculated, and thus, an approximation to the exact solution generated. For integral order solutions, one or the other of (6.1.2) is zero. For:

$$y(0) = Y_0,$$

$$y'(0) = 0,$$

the solution uses either (6.2.1) or (6.2.2), giving a complete result of:

$$\begin{aligned} y &= \alpha \operatorname{ce}_{\phi(r)}(z, q) \\ &= \alpha \sum_{r=0}^{\infty} A_{\phi(r)} \cos [\phi(r) z], \end{aligned}$$

where:

$$\begin{aligned}\phi(r) &= 2r , & (\tau = \pi) \\ &= 2r+1 , & (\tau = 2\pi)\end{aligned}$$

Applying (6.1.2) yields the final form:

$$\begin{aligned}Y_0 &= \alpha \sum_{r=0}^{\infty} A_{\phi(r)} \cos(0) , \\ \alpha &= \frac{Y_0}{\sum_{r=0}^{\infty} A_{\phi(r)}} , \\ Y &= \frac{Y_0}{\sum_{r=0}^{\infty} A_{\phi(r)}} \sum_{r=0}^{\infty} A_{\phi(r)} \cos [\phi(r)z] .\end{aligned}$$

For:

$$\begin{aligned}y(0) &= 0 , \\ y'(0) &= Y'_0 ,\end{aligned}$$

the solution is of the form of either (6.2.3) or (6.2.4), thus yielding:

$$\begin{aligned}y &= \beta \operatorname{se}_{\phi(r)}(z, q) \\ &= \beta \sum_{r=0}^{\infty} B_{\phi(r)} \sin [\phi(r)z] ,\end{aligned}$$

with:

$$\begin{aligned}\phi(r) &= 2r+2 , & (\tau = \pi) \\ &= 2r+1 , & (\tau = 2\pi)\end{aligned}$$

giving, after using (6.1.2):

$$y' = \beta \sum_{r=0}^{\infty} \{ \phi(r) B_{\phi(r)} \cos [\phi(r)z] \} ,$$

$$y'_0 = \beta \sum_{r=0}^{\infty} [\phi(r) B_{\phi(r)} \cos (0)] ,$$

$$\beta = \frac{y'_0}{\sum_{r=0}^{\infty} \phi(r) B_{\phi(r)}} ,$$

$$y = \frac{y'_0}{\sum_{r=0}^{\infty} \phi(r) B_{\phi(r)}} \sum_{r=0}^{\infty} B_{\phi(r)} \sin [\phi(r)z] .$$

D. Derivation of Piecewise Linear Solution

Since it is desired to have a piecewise linear solution of the form shown in (1.2), (6.2.1) can be approximated by:

$$y'' + p^2 y = 0$$

over a small interval, Δz . The displacement at the beginning of this interval is y_0 and the velocity y'_0 .

This equation has the following solutions (based on [34]):

A. $p^2 > 0$ [58]:

$$y = y_0 \cos pz + \frac{y'_0}{p} \sin pz .$$

B. $p^2 = 0$ [37]:

$$y = y'_0 z + y_0 .$$

C. $p^2 < 0$ [37]:

$$y = y_0 \cosh sz + \frac{y'_0}{s} \sinh sz ,$$

$$p^2 = -s^2 .$$

The final value for the characteristic number, a , and q are given such that the solution would be periodic. Δz can be of any reasonable size.

Consider now the $(m - 1)$ th and m th points. The displacements that would correspond to them are y_{m-1} and y_m , respectively. See Figure 6.1 for details. As the system moves through Δz , p^2 will change. It is better to have it as a constant value as it passes through an interval, and so, it is calculated at the midpoint of the interval, or at $(m - 1/2)\Delta z$. It is obvious that this will be the case as Δz gets smaller. That is to say, $a - 2q\cos[2(m-1)\Delta z]$ and $a - 2q\cos(2m\Delta z)$ will approach $a - 2q\cos[(2m-1)\Delta z]$.

So, for the m th point, the solutions will be as follows. For $a - 2q\cos[(2m-1)\Delta z]$ greater than zero [58]:

$$y_m = y_0 \cos p_m \Delta z + \frac{y'_0}{p_m} \sin p_m \Delta z ,$$

$$p_m = [a - 2q \cos (2m - 1) \Delta z]^{1/2} .$$

For $a - 2q\cos[(2m-1)\Delta z]$ equal to zero, the solution becomes [37]:

$$y_m = y'_0 \Delta z + y_0 .$$

And finally, for $a - 2q\cos[(2m-1)\Delta z]$ less than zero [37]:

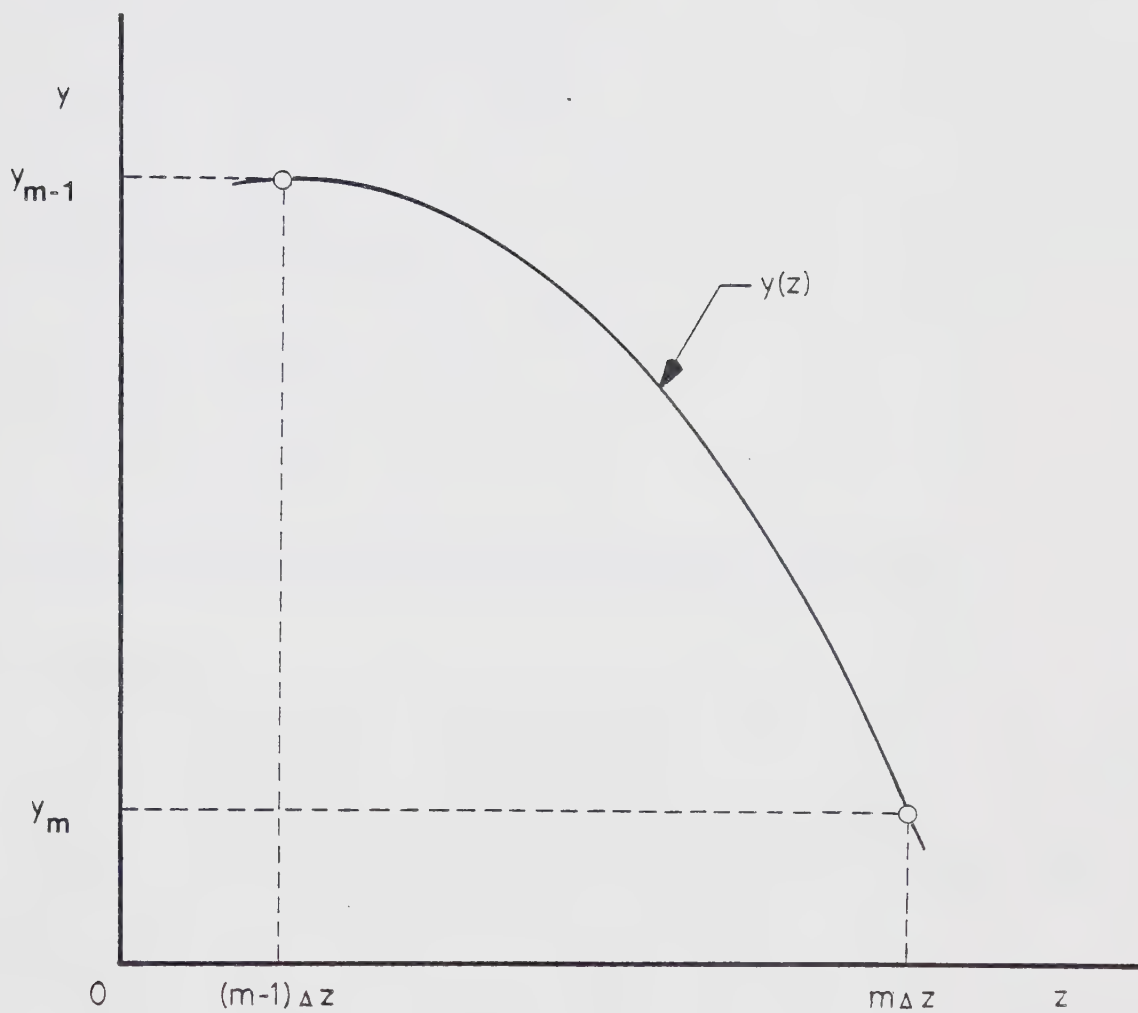


FIGURE 6.1 - TYPICAL INTERVAL LAYOUT (MATHIEU EQUATION)

$$y_m = y_0 \cosh s_m \Delta z + \frac{y'_0}{s_m} \sinh s_m \Delta z ,$$

$$s_m = \{-[a - 2q \cos (2m - 1)\Delta z]\}^{1/2} .$$

For each successive interval,

$$y_0 = y_{m-1} ,$$

$$y'_0 = y'_{m-1} .$$

At the end of the interval:

$$y_0 = y_m ,$$

$$y'_0 = y'_m .$$

and the next set of calculations for finding y_m can begin.

E. Results

Since there are numerous combinations of the characteristic number, q , and initial conditions that one can examine, it was decided to take a few selected values for the parameters and initial conditions, which would illustrate what can be encountered.

It was decided to take the initial guesses for the characteristic number as -50.0, 10.0, and 30.0, with the values for q being 1.0, 10.0, 20.0, 30.0, and 40.0. This enabled one to check some of the values obtained against tabulated results.

For the case of $q = 1.0$, both the piecewise linearization and the Runge-Kutta solution [5] had a close correspondence with the series

approximation to the exact solution, as can be seen in Figures 6.2 - 6.4. For most of the values, the same applies for $q = 10.0$ (Figures 6.5 and 6.6 being examples of the results obtained). However, something unusual occurred for the case of $q = 10.0$, $a_{\text{GUESS}} = -50.0$, $y(0) = 0.0$, $y'(0) = 1.0$ (Figure 6.7), where the piecewise linearization exhibits erratic behaviour.

Consider the cases where $q = 20.0$. Figure 6.8 is a typical example of the good correspondence between all three solutions, but Figure 6.9 shows some anomalous results. In fact, tests for this case have shown that the amplitude of the piecewise linear results can go as high as 5×10^5 , thus making the Runge-Kutta solution [5] and series approximation insignificant by comparison. A somewhat minor deviation was found to exist for the same values of a_{GUESS} and q , but for the other set of initial conditions, though this occurs near the end of the first cycle, and was less than a third of the maximum values of the other solutions. Up to that point, however, it had given satisfactory results.

Since this has been observed to happen, the question now is whether this occurs for other values of a_{GUESS} and q . Indeed it does. Compare the difference between a "good" result (that is, correspondence between all three solutions) and one that is not (Figures 6.10 and 6.11, respectively, for $q = 30.0$, and Figures 6.12 and 6.13, respectively, for $q = 40.0$).

The magnitude and form of the deviations was totally unexpected, since they did not even follow the basic pattern of the other solutions, but seem to be random occurrences, quite unlike what was seen in the previous chapter where the shape of the resulting piecewise linear solution curve at least was the same as that of the solution being

FIGURE 6.2 - SOLUTIONS TO MATHIEU EQUATION FOR

$a_{\text{GUESS}} = -50.0$, $q = 1.0$, $\Delta z = \pi/200.00$,
 $y(0) = 0.0$, $y'(0) = 1.0$

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = -50.0000
CHAR. NUM. (FINAL VALUE) = -0.1102
q = 1.0000 $\Delta z = \pi/200.0$

$y(0) = 0.0000$ $y'(0) = 1.0000$

— SERIES APPROXIMATION
○ PIECEWISE LINEARIZATION
△ RUNGE-KUTTA

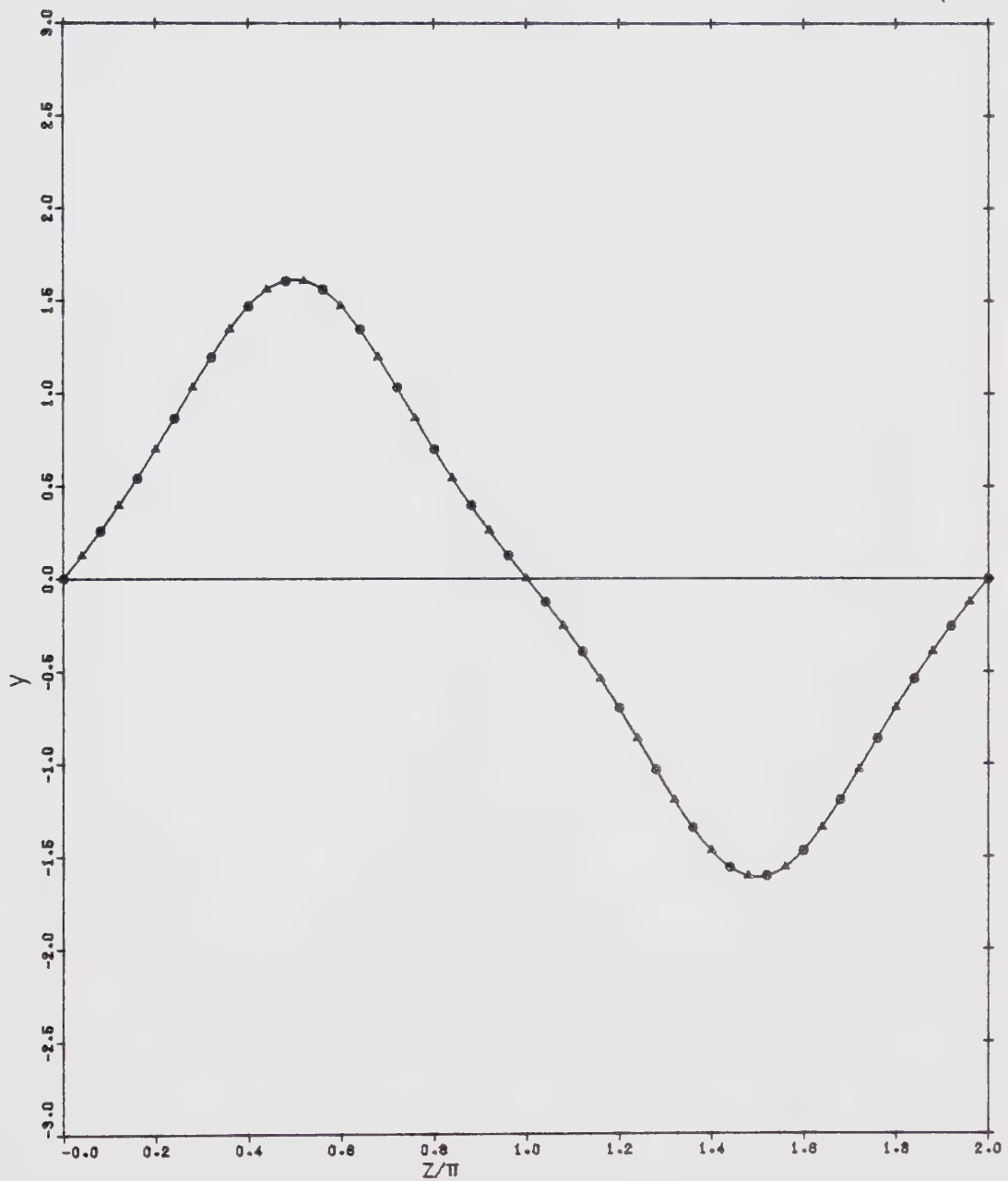


FIGURE 6.3 - SOLUTIONS TO MATHIEU EQUATION FOR

$$a_{\text{GUESS}} = 10.0, \quad q = 1.0, \quad 1x = 1/200.00,$$

$$y(0) = 1.0, \quad y'(0) = 1.0$$

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = 10.0000
CHAR. NUM. (FINAL VALUE) = 9.0784
 $q = 1.0000$ $\Delta z = \pi/200.0$

$y(0) = 1.0000$ $y'(0) = 0.0000$

— SERIES APPROXIMATION
○ PIECEWISE LINEARIZATION
△ RUNGE-KUTTA

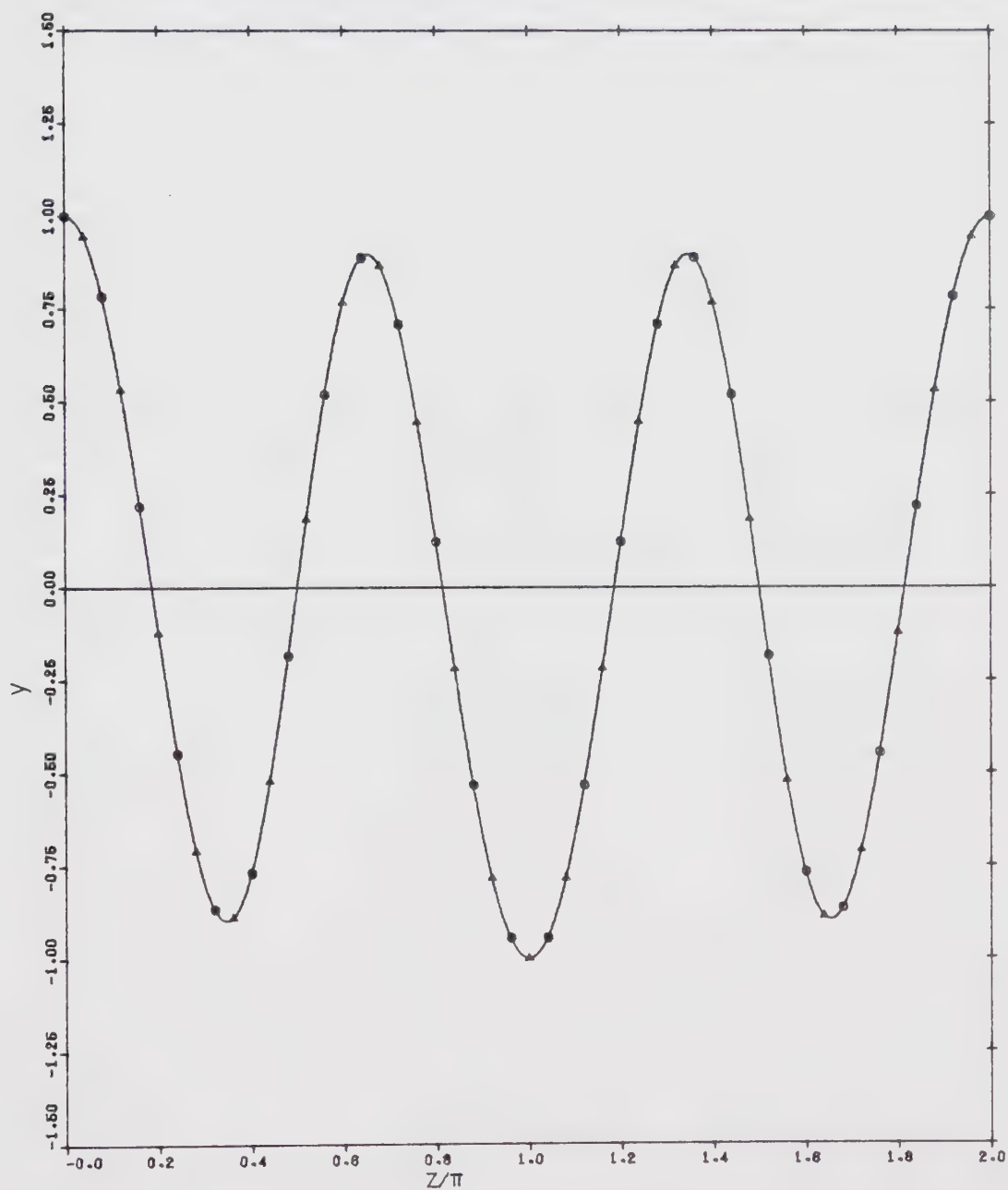


FIGURE 6.4 - SOLUTIONS TO MATHIEU EQUATION FOR
a GUESS = 30.0, q = 1.0, $\Delta z = \pi/200.00$,
y(0) = 0.0, y'(0) = 1.0

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = 30.0000
CHAR. NUM. (FINAL VALUE) = 25.0208
 $q = 1.0000$ $\Delta z = \pi/200.0$

$y(0) = 0.0000$ $y'(0) = 1.0000$

— SERIES APPROXIMATION
○ PIECEWISE LINEARIZATION
△ RUNGE-KUTTA

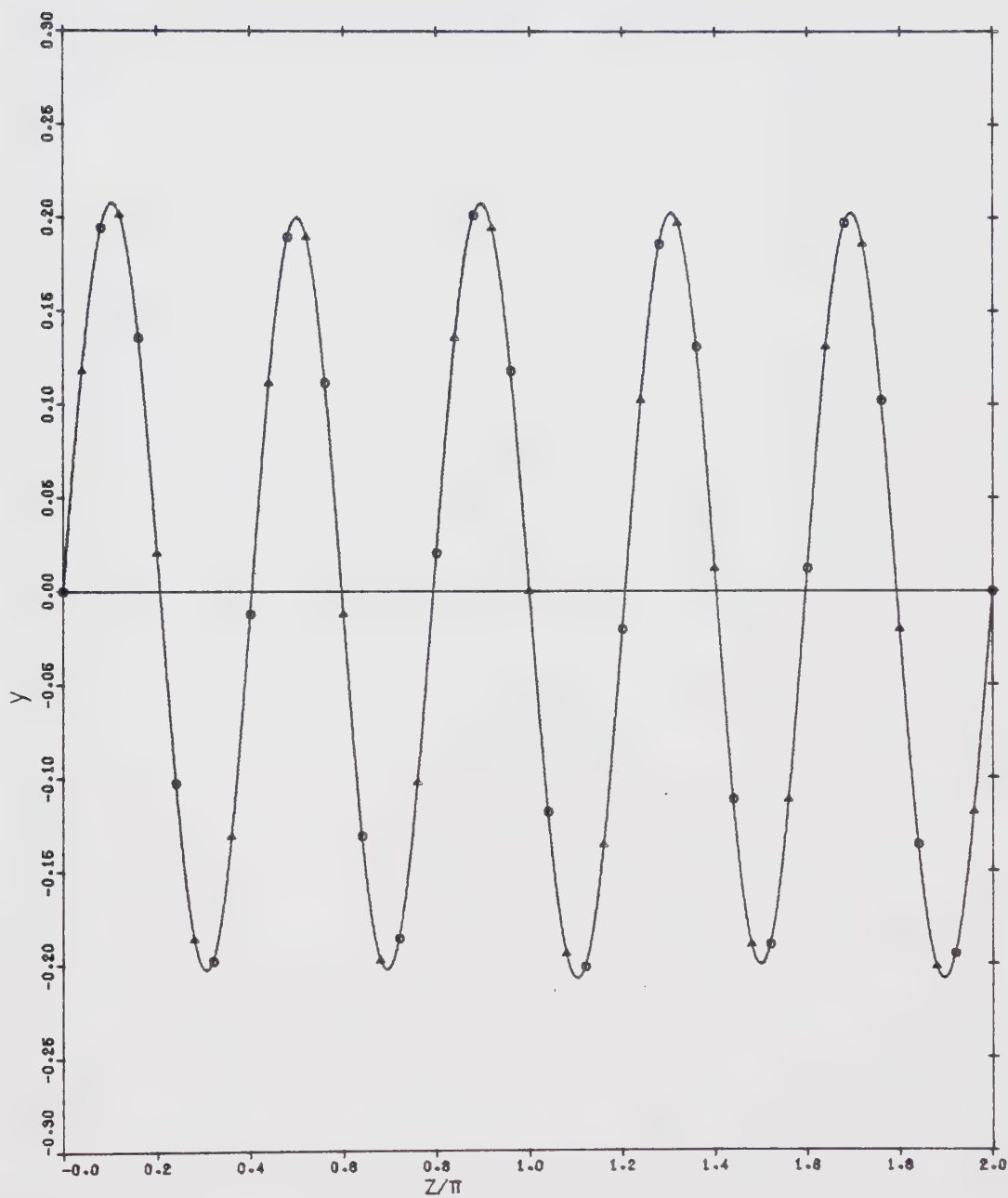


FIGURE 6.5 - SOLUTIONS TO MATHIEU EQUATION FOR
a_{GUESS} = 10.0, q = 10.0, $\Delta z = \pi/200.00$,
y(0) = 0.0, y'(0) = 1.0

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = 10.0000

CHAR. NUM. (FINAL VALUE) = 7.9861

$q = 10.0000$ $\Delta z = \pi/200.0$

$y(0) = 0.0000$ $y'(0) = 1.0000$

— SERIES APPROXIMATION

⊙ PIECEWISE LINEARIZATION

△ RUNGE-KUTTA

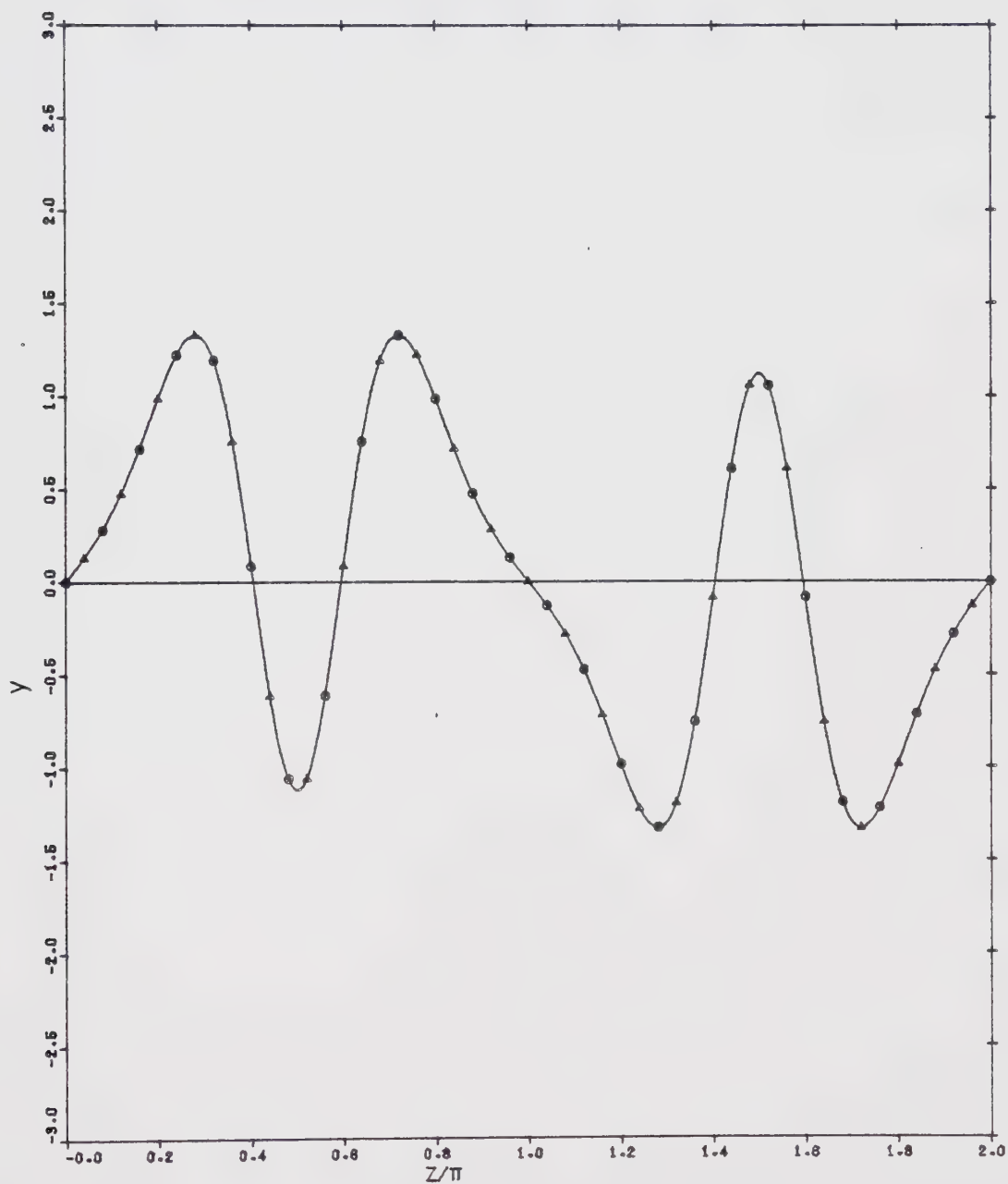


FIGURE 6.6 - SOLUTIONS TO MATHIEU EQUATION FOR
a_{GUESS} = 30.0, q = 10.0, $\Delta z = \pi/200.00$,
y(0) = 0.0, y'(0) = 1.0

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = 30.0000
CHAR. NUM. (FINAL VALUE) = 26.7664
 $q = 10.0000$ $\Delta z = \pi/200.0$

$y(0) = 0.0000$ $y'(0) = 1.0000$

— SERIES APPROXIMATION
○ PIECEWISE LINEARIZATION
△ RUNGE-KUTTA

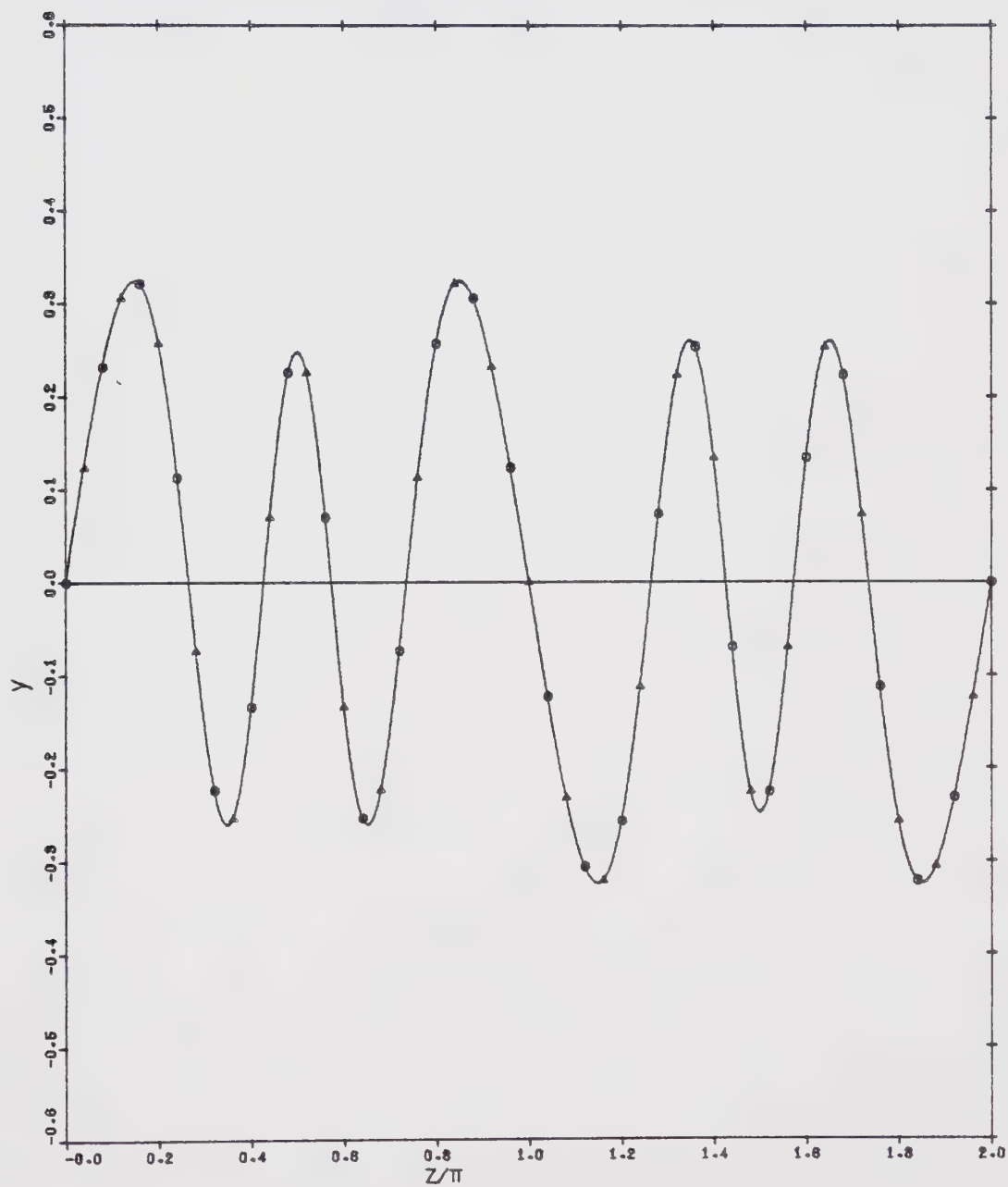


FIGURE 6.7 - SOLUTIONS TO MATHIEU EQUATION FOR
a_{GUESS} = -50.0, q = 10.0, $\Delta z = \pi/200.00$,
y(0) = 0.0, y'(0) = 1.0

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = -50.0000
 CHAR. NUM. (FINAL VALUE) = -13.9365
 $q = 10.0000$ $\Delta z = \pi/200.0$

$y(0) = 0.0000$ $y'(0) = 1.0000$

— SERIES APPROXIMATION
 ○ PIECEWISE LINEARIZATION
 △ RUNGE-KUTTA

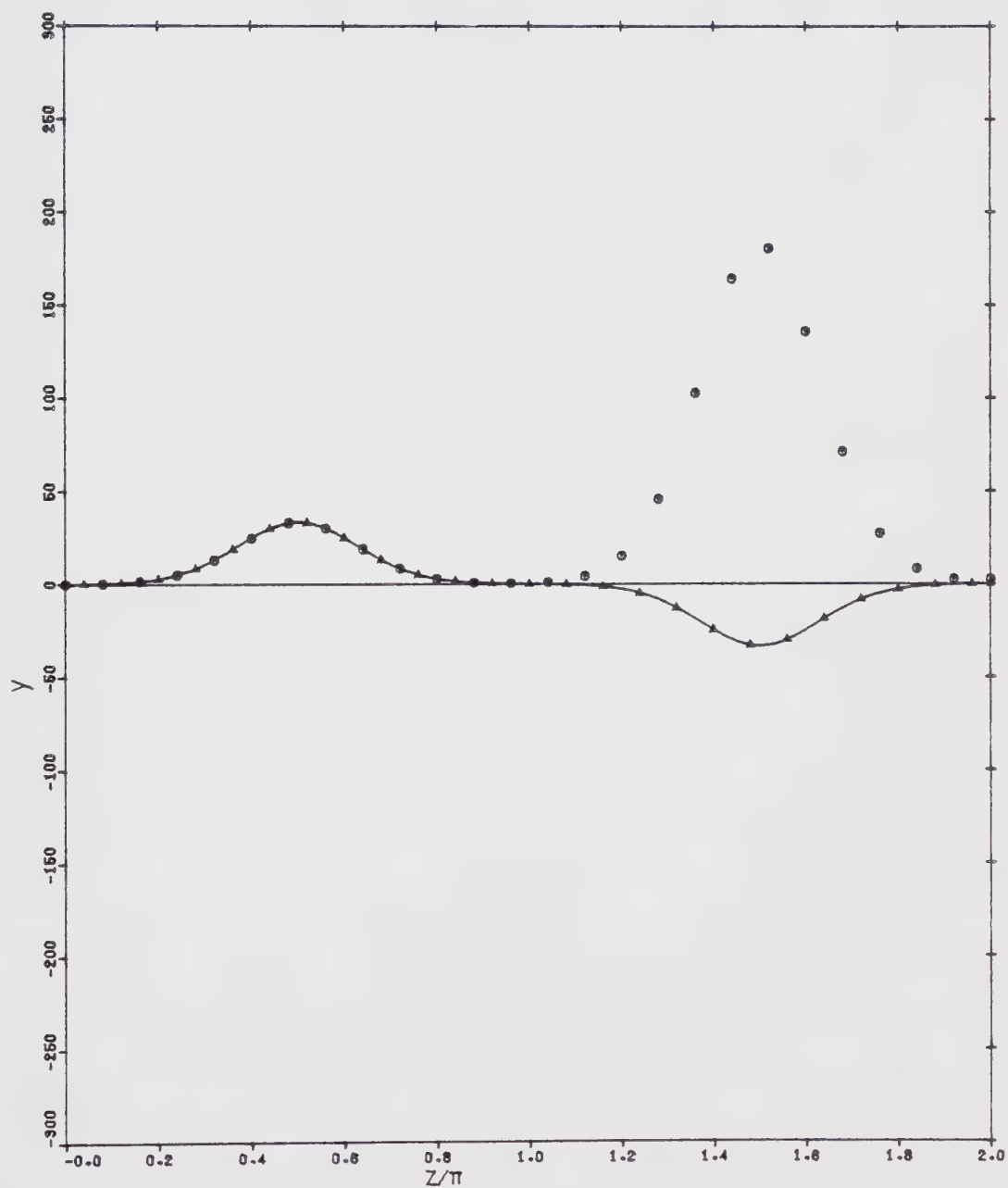


FIGURE 6.8 - SOLUTIONS TO MATHIEU EQUATION FOR
a_{GUESS} = 10.0, q = 20.0, $\Delta z = \pi/200.00$,
y(0) = 1.0, y'(0) = 0.0

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = 10.0000
 CHAR. NUM. (FINAL VALUE) = 15.3958
 $q = 20.0000$ $\Delta z = \pi/200.0$

$y(0) = 1.0000$ $y'(0) = 0.0000$

— SERIES APPROXIMATION
 ○ PIECEWISE LINEARIZATION
 ▲ RUNGE-KUTTA

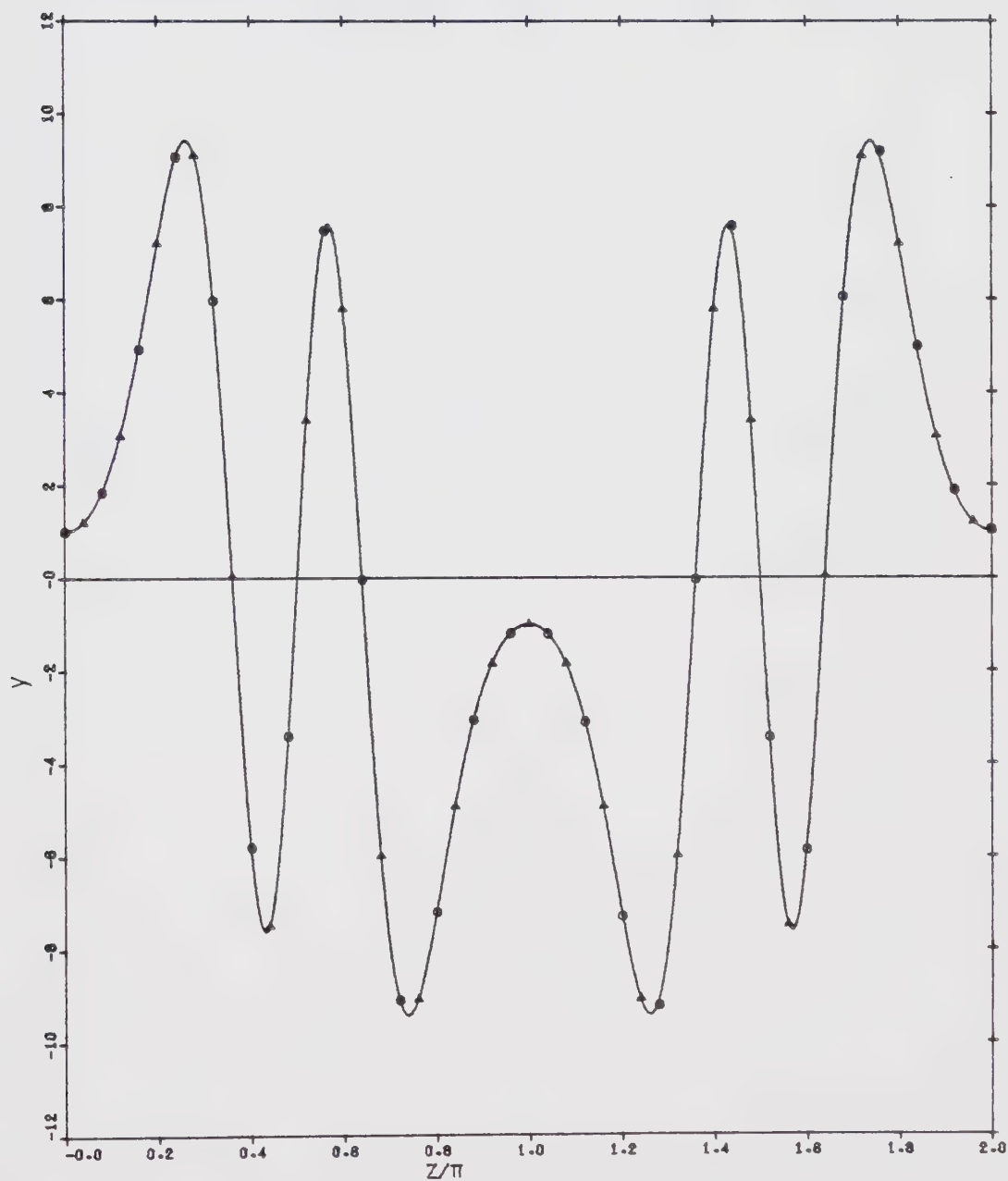


FIGURE 6.9 - SOLUTIONS TO MATHIEU EQUATION FOR
a GUESS = -50.0, q = 20.0, $\Delta z = \pi / 200.00$,
y(0) = 0.0, y'(0) = 1.0

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = -50.0000
 CHAR. NUM. (FINAL VALUE) = -31.3134
 $q = 20.0000$ $\Delta z = \pi/200.0$

$y(0) = 0.0000$ $y'(0) = 1.0000$

— SERIES APPROXIMATION
 ○ PIECEWISE LINEARIZATION
 △ RUNGE-KUTTA

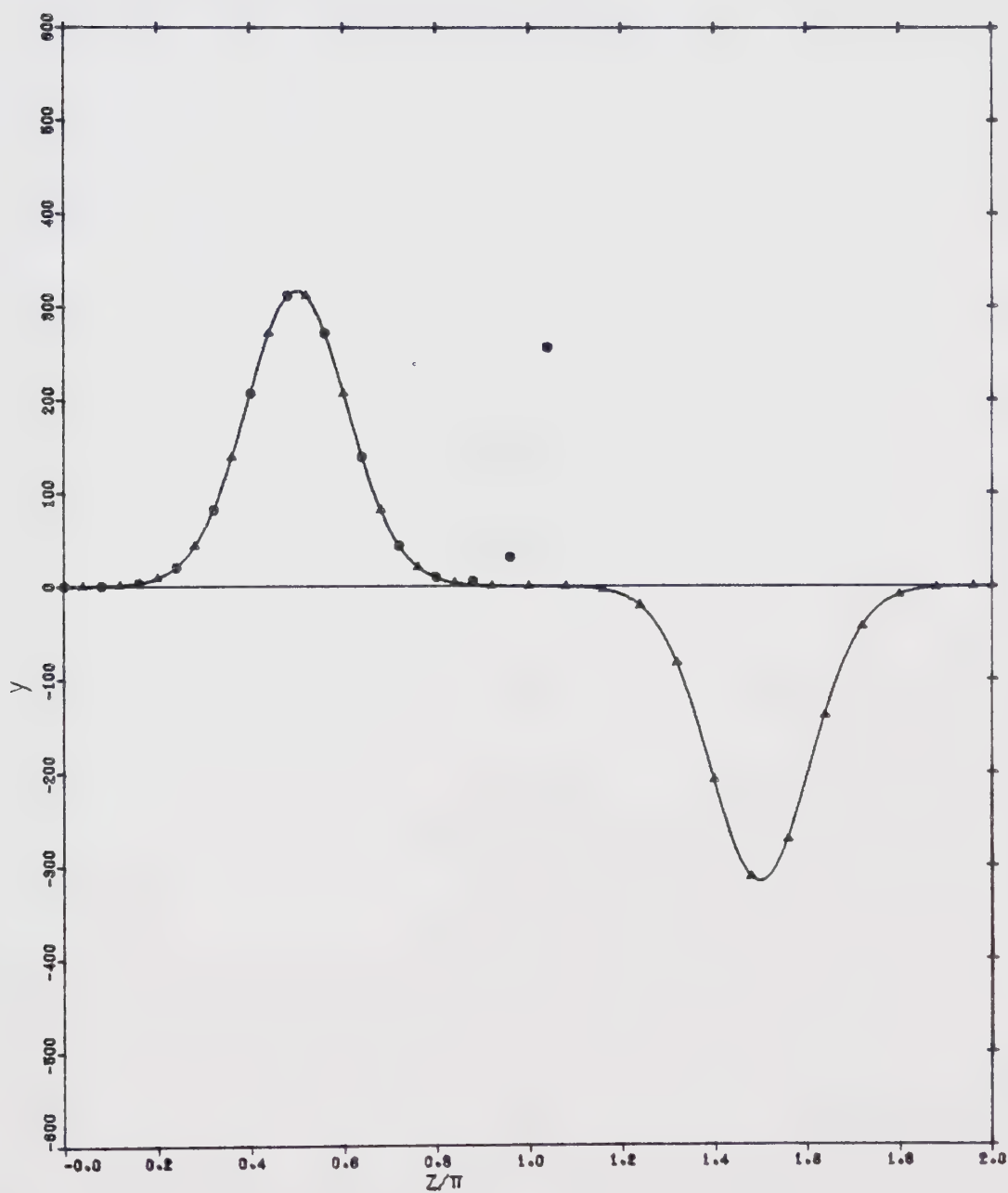


FIGURE 6.10 - SOLUTIONS TO MATHIEU EQUATION FOR
a_{GUESS} = 30.0, q = 30.0, $\Delta z = \pi/200.00$,
y(0) = 0.0, y'(0) = 1.0

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = 30.0000
CHAR. NUM. (FINAL VALUE) = 26.8133
 $q = 30.0000$ $\Delta z = \pi/200.0$

$y(0) = 0.0000$ $y'(0) = 1.0000$

— SERIES APPROXIMATION
○ PIECEWISE LINEARIZATION
△ RUNGE-KUTTA

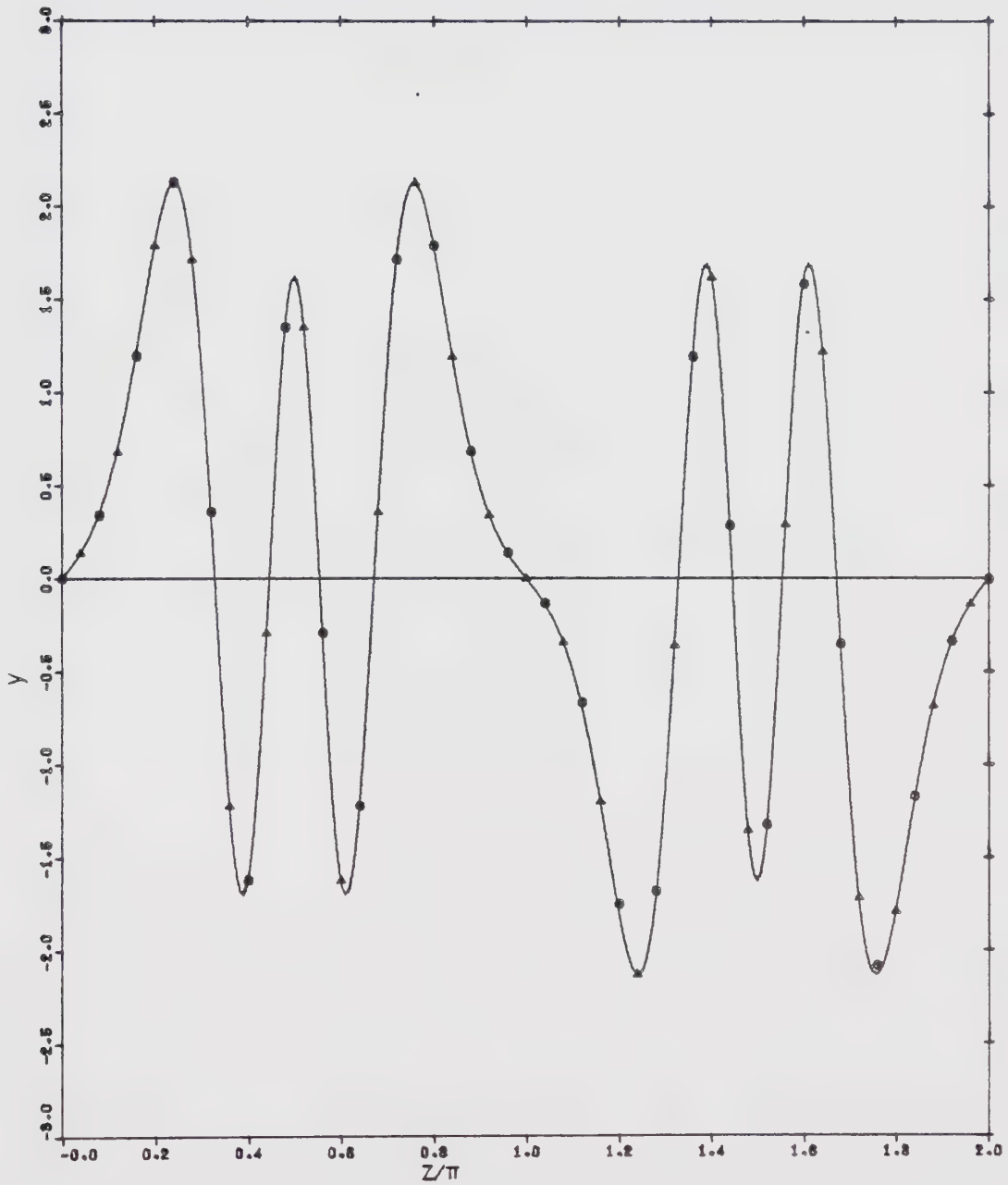


FIGURE 6.11 - SOLUTIONS TO MATHIEU EQUATION FOR
a = -50.0, q = 30.0, $\Delta z = \pi/200.00$,
GUESS y(0) = 0.0, y'(0) = 1.0

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = -50.0000
 CHAR. NUM. (FINAL VALUE) = -49.3016
 $q = 30.0000$ $\Delta z = \pi/200.0$

$y(0) = 0.0000$ $y'(0) = 1.0000$

— SERIES APPROXIMATION
 ○ PIECEWISE LINEARIZATION
 ▲ RUNGE-KUTTA

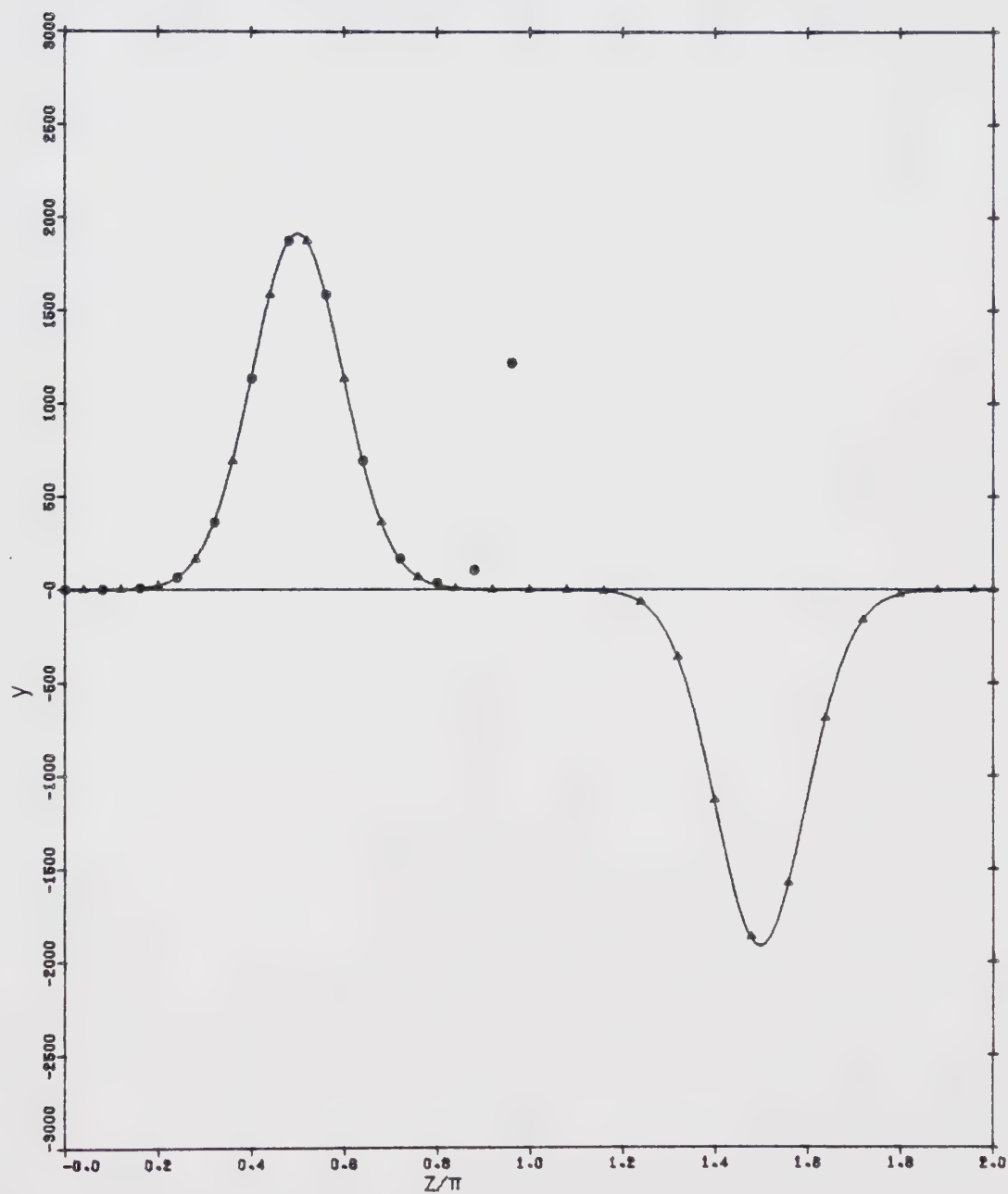


FIGURE 6.12 - SOLUTIONS TO MATHIEU EQUATION FOR
a GUESS = 30.0, q = 40.0, $\Delta z = \pi/200.00$,
y(0) = 1.0, y'(0) = 0.0

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = 30.0000
CHAR. NUM. (FINAL VALUE) = 22.3253
 $q = 40.0000$ $\Delta z = \pi/200.0$

$y(0) = 1.0000$ $y'(0) = 0.0000$

— SERIES APPROXIMATION
○ PIECEWISE LINEARIZATION
△ RUNGE-KUTTA

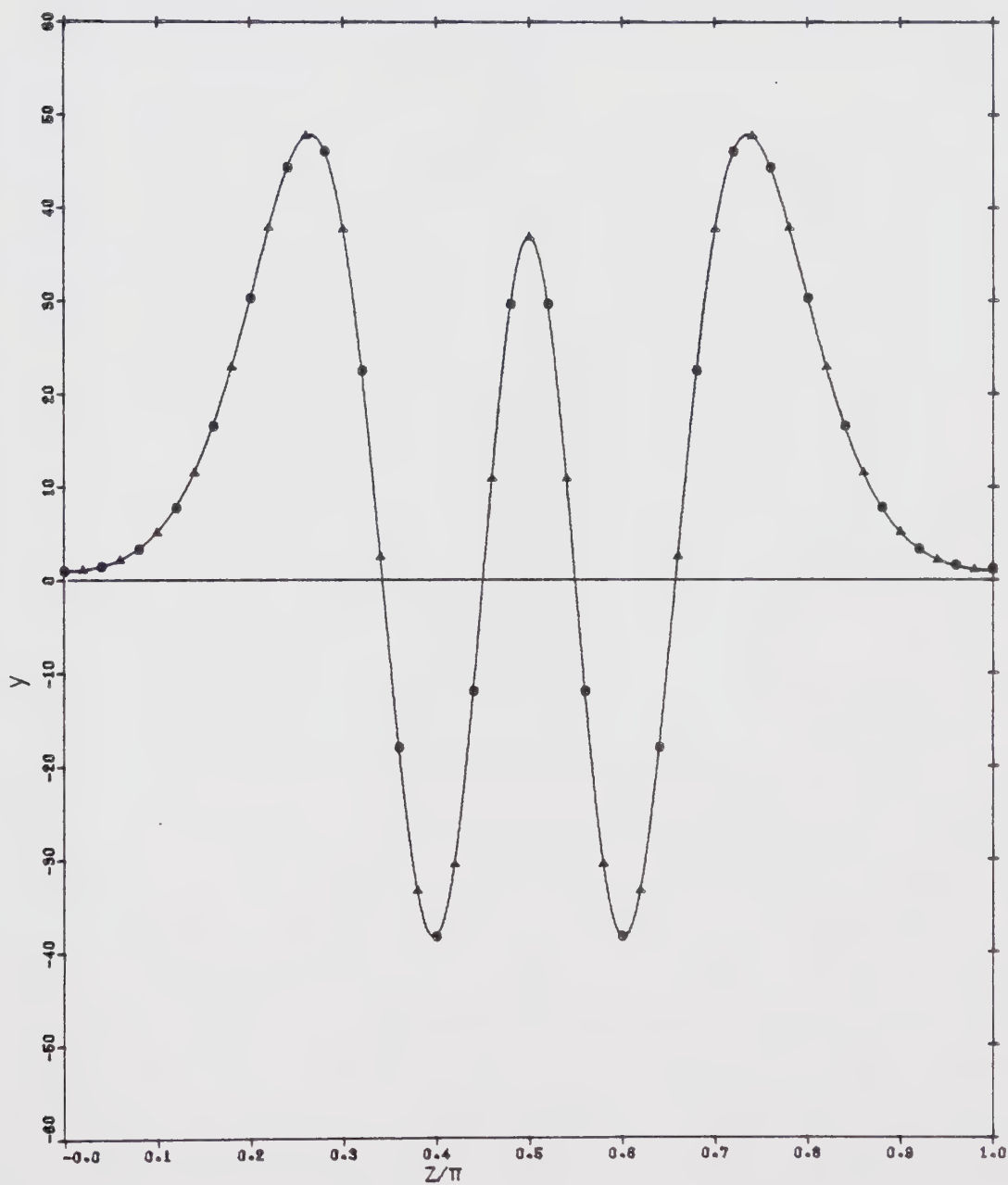


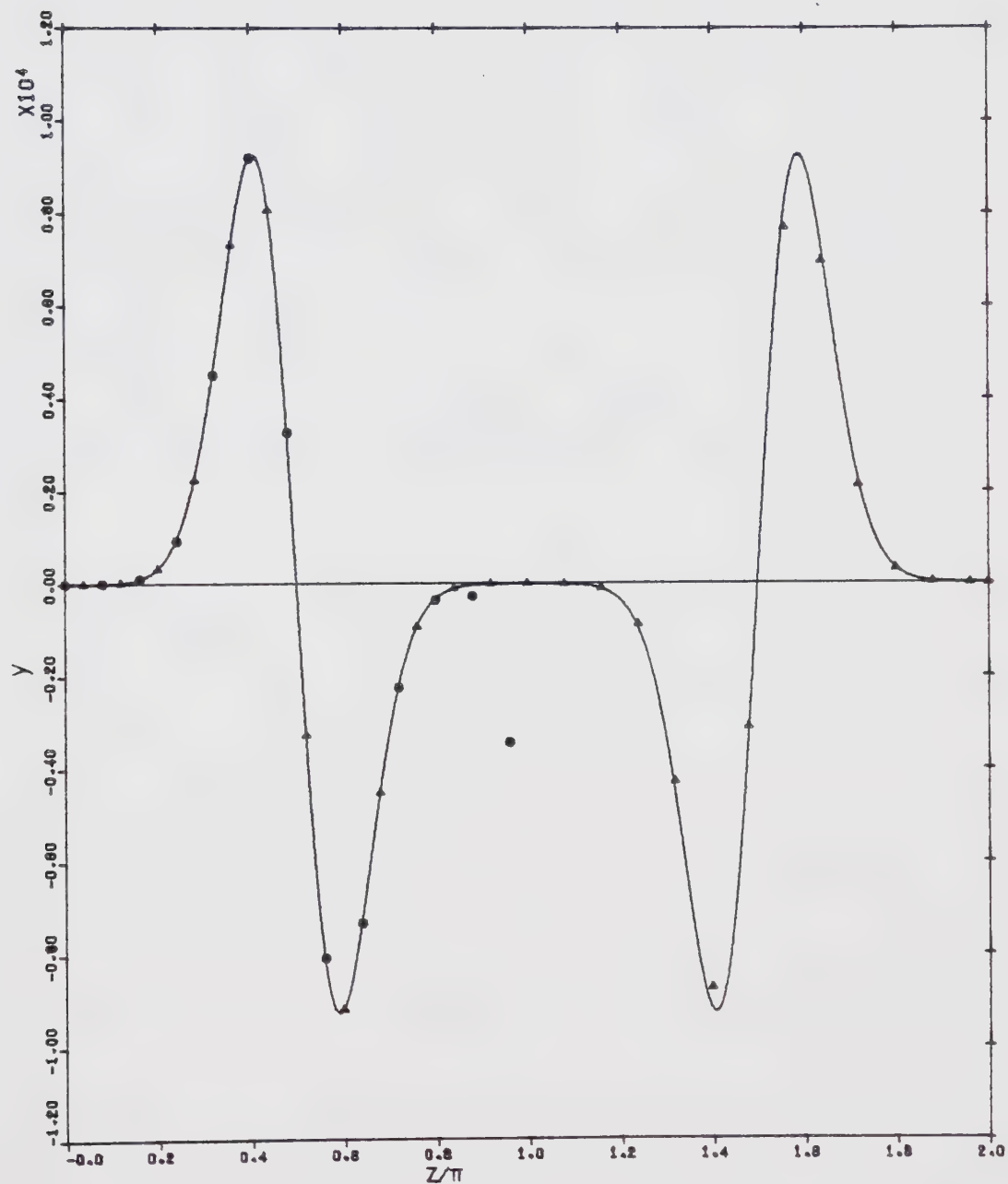
FIGURE 6.13 - SOLUTIONS TO MATHIEU EQUATION FOR
a = -50.0, q = 40.0, $\Delta z = \pi/200.00$,
GUESS y(0) = 1.0, y'(0) = 0.0

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = -50.0000
 CHAR. NUM. (FINAL VALUE) = -43.3522
 $q = 40.0000$ $\Delta z = \pi/200.0$

$y(0) = 1.0000$ $y'(0) = 0.0000$

— SERIES APPROXIMATION
 ○ PIECEWISE LINEARIZATION
 ▲ RUNGE-KUTTA



compared. Since changing the size of the increment of z seemed to work for the Van der Pol equation, it was felt that this may solve the problem. A typical result of that was as can be seen in Figure 6.14. This particular case was chosen as this phenomenon was first observed during the initial development of the computer program using the same conditions, but a a_{GUESS} being -45.0 .

It was suggested that the computer program written to generate the solutions and plots could be run up to a point just prior to the breakdown, and then using the final values as a new set of initial conditions. Unfortunately, this helped very little. As a check as to whether the computer program was at fault, a test program was run on a Hewlett-Packard HP-67 hand calculator using the same conditions, and it was found that the same situation occurred with essentially the same outcome. This was double-checked by running a Runge-Kutta routine designed for the HP-67 by Hewlett-Packard [59], but with the same result.

At present, the question as to what occurred, and why, remains unanswered. However, it would appear from the plots obtained for the region of $q \leq 20.0$ that no problems would be experienced for a positive value of the characteristic number. Possibly, a small negative value might do so, but this was not investigated. There are indications that this may even extend up to $q = 30.0$. Problems quite definitely develop for $q = 40.0$, and it appears that regardless of what the value for the characteristic number is, deviations from the series approximation will occur.

There does not appear to be any symmetry in the severity of the deviations as far as initial conditions are concerned. An example of

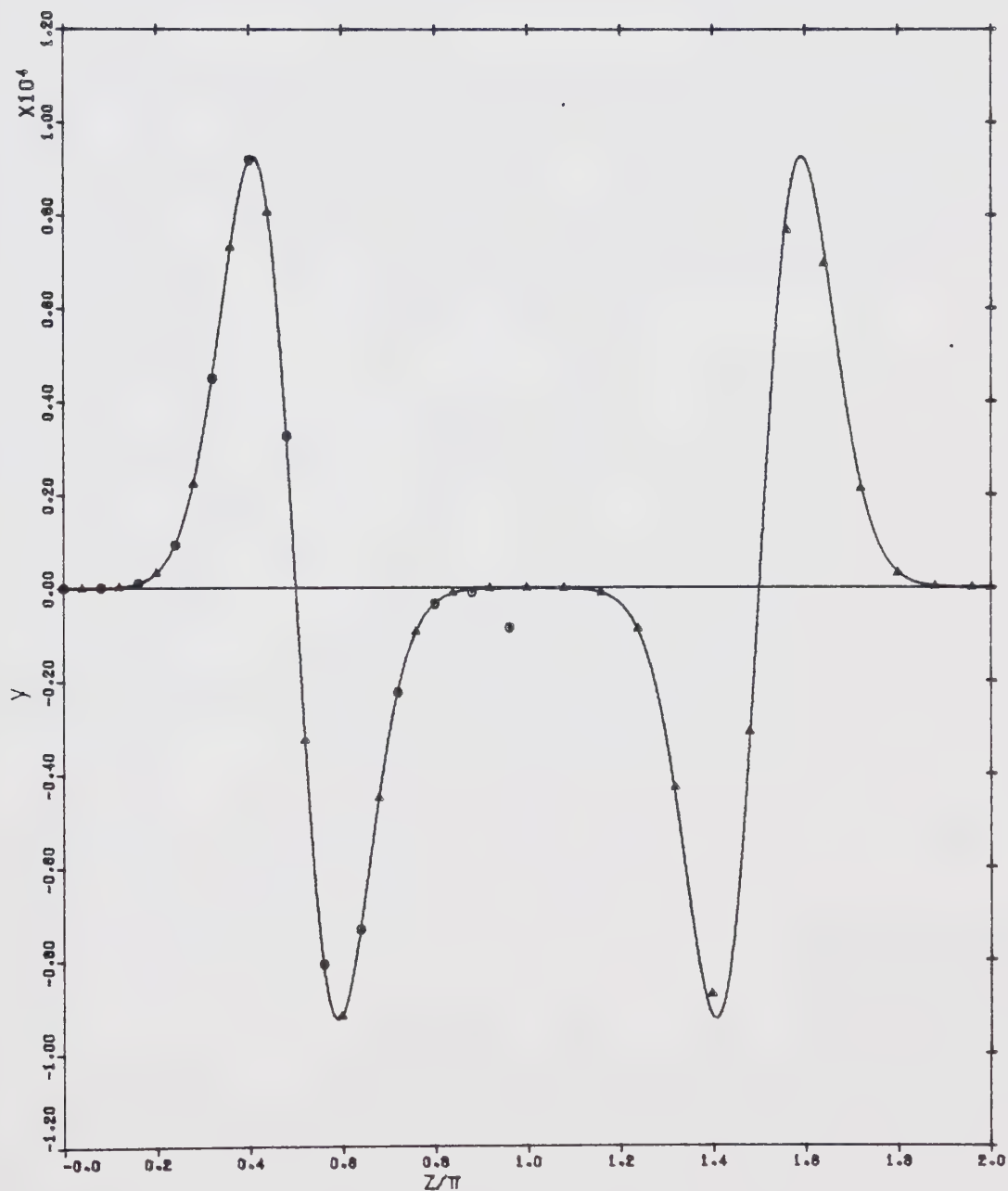
FIGURE 6.14 - SOLUTIONS TO MATHIEU EQUATION FOR
a_{GUESS} = -50.0, q = 40.0, $\Delta z = \pi/400.00$,
y(0) = 1.0, y'(0) = 0.0

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = -50.0000
 CHAR. NUM. (FINAL VALUE) = -43.3522
 $q = 40.0000$ $\Delta z = \pi/400.0$

$y(0) = 1.0000$ $y'(0) = 0.0000$

— SERIES APPROXIMATION
 ○ PIECEWISE LINEARIZATION
 △ RUNGE-KUTTA



this is the case of $a_{\text{GUESS}} = -50.0$ and $q = 10.0$. For the conditions of $y(0) = 1.0$, and $y'(0) = 0.0$, there is a slight deviation near the end of the first cycle, but only a fraction of the maximum displacement. However, for the same characteristic number estimate and q , but $y(0) = 0.0$ and $y'(0) = 1.0$, a totally different situation arises in which the maximum displacement for the piecewise linearization dwarfs the other solutions. Though this is an isolated example, there may possibly be a problem here.

As had been shown in the previous chapters, the program had been stripped to its essential calculations, removing comment and output statements, and the timings of the three solutions were obtained [27] for selected cases. The results of this can be seen in Figures 6.15 - 6.17. This was run on the Amdahl 470V/8 [26].

For the piecewise linearization and the series approximation, a linear relationship with the number of divisions of π (and hence, the size of the increment of z) and the execution time exists, but the Runge-Kutta method [5] displays some unusual features. First, its execution times are almost always larger than those for the other two solutions, except for Figure 6.15 and the smaller values of Δz in Figure 6.16. In fact, it is considerably larger than what was obtained for the piecewise method, being larger by about a factor of 10 for a large Δz but down to four times as large for the smaller values.

The other unusual feature is the fact that the Runge-Kutta results [5] show some scatter, particularly seen on Figure 6.16. This is different than what was observed for the hard spring equations (in which the results were linear) and for the Van der Pol equation where the results were exponential.

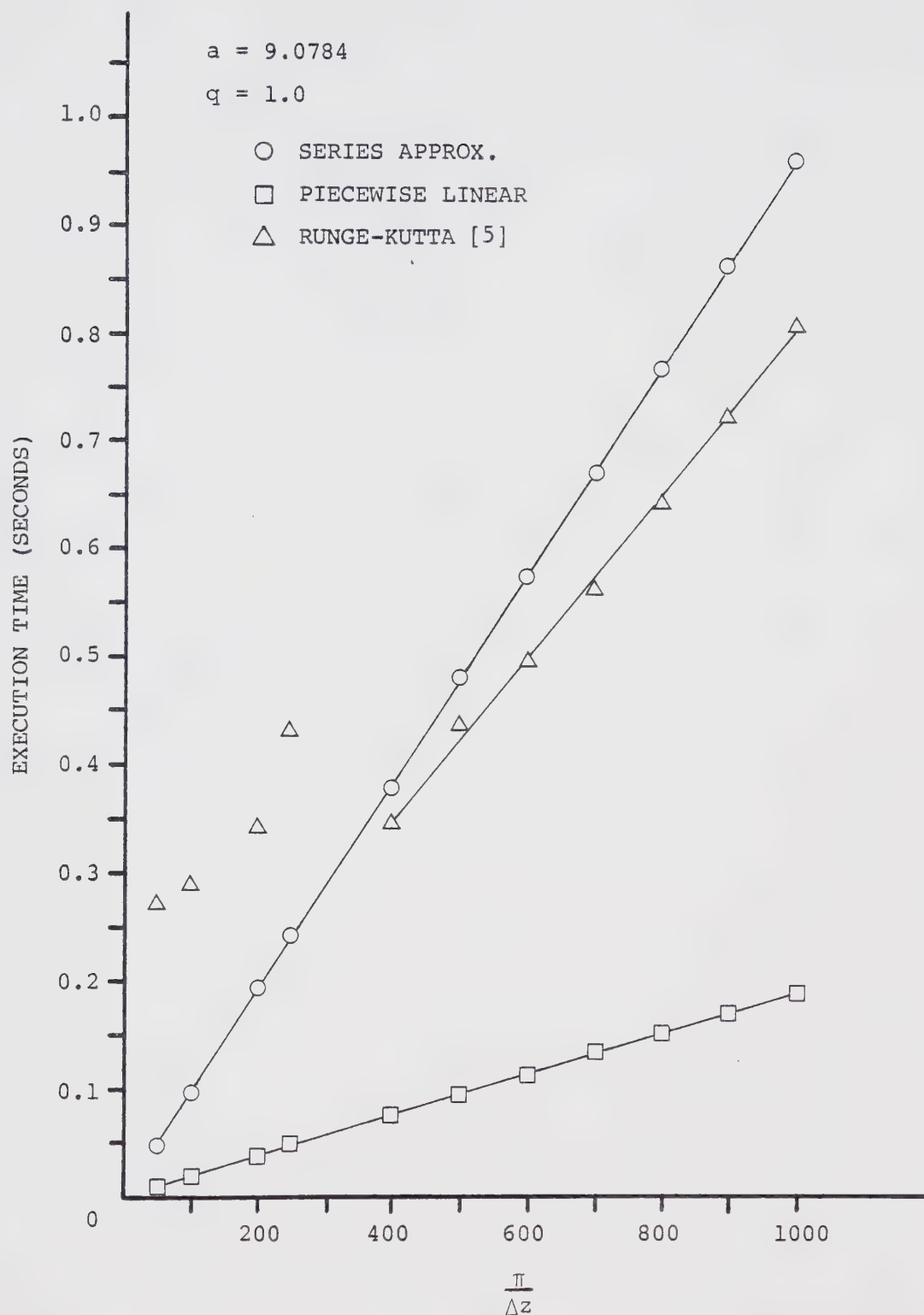


FIGURE 6.15 - EXECUTION TIMES FOR SOLUTIONS TO
 MATHIEU EQUATION FOR $a_{\text{GUESS}} = 10.0$, $q = 1.0$,
 $y(0) = 1.0$, $y'(0) = 0.0$ [27]

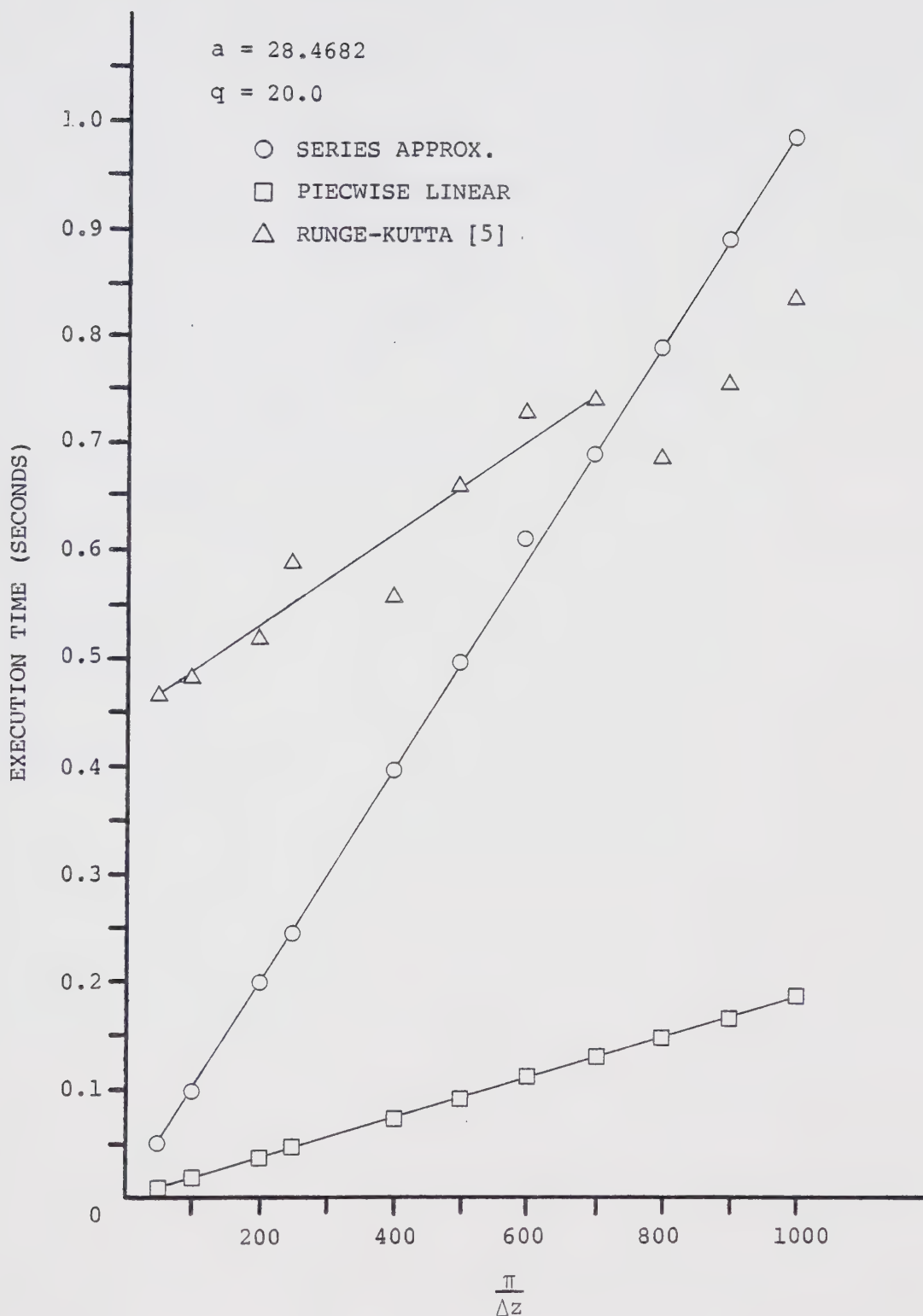


FIGURE 6.16 - EXECUTION TIMES FOR SOLUTIONS TO
 MATHIEU EQUATION FOR $a_{\text{GUESS}} = 30.0$, $q = 20.0$,
 $y(0) = 1.0$, $y'(0) = 0.0$ [27]

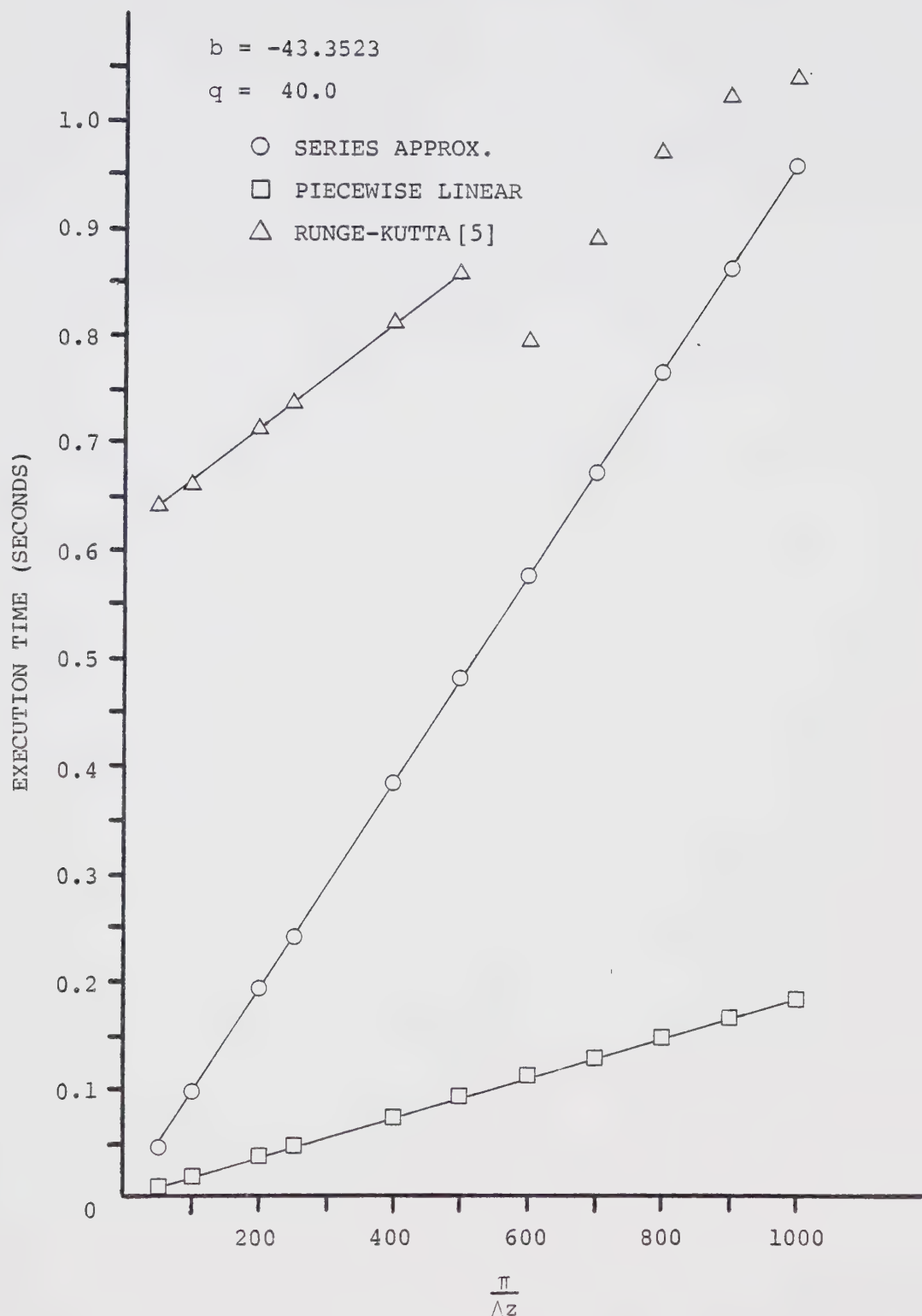


FIGURE 6.17 - EXECUTION TIMES FOR SOLUTIONS TO
 MATHIEU EQUATION FOR $a_{\text{GUESS}} = -50.0$, $q = 40.0$
 $y(0) = 1.0$, $y'(0) = 0.0$ [27]

This behaviour was discussed with a consultant from the University of Alberta Computing Services [60] in an attempt to determine what had brought this about. A suggested explanation is that the routine that generated the Runge-Kutta results [5] may have been affected in whole or in part by the actual size of the Δz used. However, the exact cause of the execution time variation remains unknown.

VII. DISCUSSION

As was mentioned in Chapter 2, the method of tangents for solving the undamped hard spring equation had a greater execution time than the method of chords because of the extra calculations that had to be carried out in order to arrive at a solution. The period of oscillation as calculated by the former method was closer to the value obtained by the approximation to the exact solution [4], [15], [17] than the period calculated by the latter one, but the method of chords had a closer fit to the solution curve of this approximation than did the one using tangents.

For 5 segments, the period as calculated by the approximation to the exact solution [4], [15], [17] for $a = 1.0$ and $b = 2.0$ is 4.004308722, while the one obtained by chords is 3.976017631 with the one by tangents, 4.018171439. The period found by the method of chords was 0.707% less than (2.2.5) [17], while the value for the method of tangents was 0.346% larger. If one were to consider the case of 100 segments for $a = 1.0$ and $b = 0.15$, where (2.2.5) [17] was found to be 5.958299670, the chords gave 5.958281464 (0.0003% less) while the tangents gave 5.958310457 (0.0002% greater). So, the tangents gave a more accurate calculation, as far as the magnitude of error is concerned, for the period.

However, in judging a method of solution, the period itself is not necessarily the only standard. The accuracy of the fit of the calculated solution with respect to the one that had been chosen to be the benchmark [4], [15], [17] is perhaps more important, and it is here that the method of tangents starts showing a small, but significant,

difference. If one were to look at Figure 2.8, one can see that the tangents are off by a slight amount, while the chords are much closer. For Figures 2.6 and 2.7, this does not become immediately apparent, which indicates that the values of a and b would have some bearing on the closeness of the fit.

As mentioned before, it was because the method of chords gave a closer fit that this method was adopted throughout the remainder of the thesis as the method of piecewise linearization, as well as the fact that it was quicker to calculate.

The execution time provided some interesting results. Since the solution chosen to be the standard was based on a packaged program library [5], one would have expected it to have a shorter execution time than the other solutions, but since (2.2.4) [15] was calculated each time, this apparently unusual result appears to be resolved. It should be noted that while setting up the timing program, the author had run it in which (2.2.4) [15] was calculated only once and the solutions generated after that, and the approximation to the exact solution did, in fact, turn out to be faster by about 10% than that of the method of chords for most of the runs.

The two cases of the damped hard spring were an extension of the undamped one. For the situations examined, a good agreement between the results of the piecewise solution and those of the Runge-Kutta method [5] was obtained, which leads one to conclude that for the size of the displacement used, the piecewise linearization was a suitable approximation to the chosen standard, since an exact solution is not available.

Because the same calculations are carried out for each iteration, the timing results (using [27]) were linear with respect to the number

of divisions per first quarter-cycle will occur, and also, the quantities being incremented were different in each solution. Also, as expected, the Runge-Kutta method [5] used less time than the piecewise method. This can be seen if one considers that this is from a packaged program library, and so, should have the advantage of development and modifications. In fact, the IMSL package used was in its seventh edition [5], and presumably, should be as efficient as possible, as compared to the piecewise linearization portion of the computer program used to generate the solutions.

What is unusual is that for a large increment (that is, a small number of divisions per first quarter-cycle), the timings show a large deviation, which probably means that the increment chosen was so large that, in fact, the solution does not have time to converge (if a time was obtained) or that both solutions have to work harder to achieve a result. This appears further on for the Van der Pol equation.

The hard spring equation piecewise linear solutions gave a close correspondence with the results from the Runge-Kutta method [5]. However, this was not the case for the piecewise linearization of the Van der Pol equation and the same Runge-Kutta method [5], and for the piecewise linear results for the Mathieu equation and the approximation to the exact solution.

The apparent "phase difference" noticed in the Van der Pol equation results, and which apparently was resolved by reducing the size of the time increment used, was the first sign that the piecewise linearization could have some limitations. Some insight was gained by considering the subject of stability, since this may have some bearing on the results obtained for the Mathieu equation.

Several things should be noted first before continuing. One is that because of the apparently strange results obtained for the Van der Pol equation for the higher values of μ , there may be a limit beyond which the piecewise linearization becomes unreliable. In other words, for the equation parameters and increment sizes, there may be an area in which this method is ineffective. This is indicated by Figures 5.21 and 5.22 [38], [39].

If this is the case, then, can one predict if, and when, this will occur for a particular situation? Is it actually due to the numerical instability or possibly machine error? The answer to the first question is that it may not always be possible to do so. This is indicated by Greenspan [61], and it would therefore appear that a trial-and-error approach is the only sure method of determining if an equation becomes unstable for a given set of parameters.

The question now is whether this error was caused by instability or machine error. The internal tolerance for calculations like the Newton-Raphson [30] relation that was used for determining the remaining time until a peak or trough had been reached had been set to smaller values to ensure that perhaps this may not have been the source. This was because the solutions appeared to be all right for the smaller values for μ . Also, the tolerance for the Runge-Kutta program had also been reset to a smaller value [5], but the result was that virtually no change was noticed in this solution. Since the smaller value for the time increment appeared to resolve the problem, it was concluded that numerical instability may have caused it.

This led to the consideration of the accuracy of the solution to (5.1.1) obtained by using the Runge-Kutta method [5]. Figures 5.21 and

5.2.2 [38], [39] indicated that it may become inaccurate with a large time interval, but perhaps not to the same extent as the piecewise linearization. It can therefore be concluded that the results from the Runge-Kutta method [5] may be misleading if factors like this are not taken into account.

Next, the timing runs were done [27]. The piecewise linearization had a parabolic relationship between execution time and the size of the time increment, and the Runge-Kutta [5] had an exponential relation. This can be easily explained by referring for a moment back to the results for the damped hard spring equations.

There, for a small number of divisions for the first quarter-cycle, the times for both solutions deviated significantly from the remainder of the results, which indicated that both solutions had to go through more iterations at this stage. The same probably occurs here, since there were some runs for which times were not obtained because the computer kept exceeding its set time limit [31]. This occurred for the higher values of Δt_P and Δt_{RK} . For a small value of Δt a large number of iterations had to be carried out, though, because of the small increment, convergence would be rather quick [60].

The aspect of instability occurs once again with the Mathieu equation, and this will now be examined. For the lower values of q , there was generally no problem with the characteristic number was large and negative, and what makes it particularly puzzling is the fact that the situation described by Figure 6.7 does not seem to follow any particular pattern relating to the other solutions. For the Van der Pol equation, the deviations at least looked like the Runge-Kutta solution [5], but expanded along the time axis. However, for the Mathieu

equation, this does not seem to happen.

Figure 6.7 may very well represent part of the unstable region, as had just been discussed. Figures 6.9, 6.11, 6.13, and 6.14 were interesting in the fact that the maximum amplitude of the piecewise linearization completely dwarfed the other solutions as was seen when the initial results were obtained. (It should be noted that the plots were scaled by taking the largest amplitude of the three solutions, and so, this result will occur.) The program was temporarily modified to show the other solutions, and so, the piecewise linearization results end up going off scale.

In particular, Figures 6.13 and 6.14 display that changing the size of Δz does not seem to have any major effect on this phenomenon. The analysis is further complicated through the piecewise linear results following the series approximation to the exact solution quite closely until nearly the end of the first half-cycle, and then becoming unstable.

Though not shown here, it should be noted that the results from the Runge-Kutta method [5] had exhibited similar behaviour. This was observed by the author when developing the program for solving the Mathieu equation for the same initial conditions for Figures 6.13 and 6.14, but using an a_{GUESS} of -45.0 instead of -50.0. It became unstable to a smaller extent than the piecewise linearization and took place in roughly the same area, while the series approximation was "well-behaved".

Perhaps some insight can be gained by examining the subject of numerical instability. First, one must consider what sort of instabilities exist.

Most texts on numerical analysis will give four definitions, depending upon what the situation in question is, such as [62], [63], [64], [65], [66]. (It should be noted that the definitions to be given are as used in the references.)

"Inherent" instability occurs when a small change in the starting value will cause large changes in the solution to occur, but this is independent of the numerical method chosen to solve the problem [62]. It is the equation itself that is insensitive, and not the means used to solve it [63].

"Partial" instability arises when the solution deviates significantly from the true one as more steps are taken. The situation is dependent upon the equation that is being solved, the method used, and especially the step size. This appears to be resolved when a smaller increment is chosen [64].

"Strong" instability arises when one solves the equation, but additional terms arise that do not vanish as the step size gets very small. These terms increase faster than the desired solution, and lack of convergence as well as stability is implied here [65], [66].

"Weak" instability is defined as having the same behaviour as what characterizes strong instability occurs, but the solution converges [65].

In view of these definitions, the problems with the Mathieu equation piecewise linear solution seem to display a combination of all of these, but cannot be completely described by any one of them. As was mentioned before, the initial conditions were changed by a slight amount, but this did not give rise to the radical changes that "inherent" instability would imply. "Partial" instability [64] seems to

describe what had occurred with the Van der Pol equation, but still, a change in the step size did not seem to matter much, though there is some indication that if one were to take an extremely small step size (such as on the order of $\pi/5000$), a partial success is achieved in that the amplitude of the piecewise linear method is reduced by at least a factor of 10, but the "takeoff" point for this irregularity does not seem to have been affected much by being delayed for a few more increments, as compared to what happened with the Van der Pol equation. So, it would seem that "partial" instability [64] can be eliminated as a probable cause.

After breakdown, the solution appears to follow a pattern in the resulting curve rather than having the points randomly distributed. This may be an indication of "weak" instability [65], but because there was a good correspondence between the piecewise linearization and the series approximation to the exact solution, it cannot be the sole cause. The possible clue concerning "weak" instability [65] was obtained from the original plot from which Figure 6.9 was derived (Figure 7.1)

Since no single definition of instability fully describes the observed results, it appears that the reasons for the failure of the piecewise linear method are perhaps more complex than first imagined.

Figure 7.2 is a plot of the characteristic numbers obtained versus q , and the distinction is made between those values for which serious deviations occur and those for which the piecewise linear solution is reasonably well-behaved. There appears to be a transition zone between which the solutions go from stable to unstable. Unfortunately, there is not enough data available to give a better resolution of the boundaries, and so, Figure 7.2 has to be taken as a very rough estimate. What it

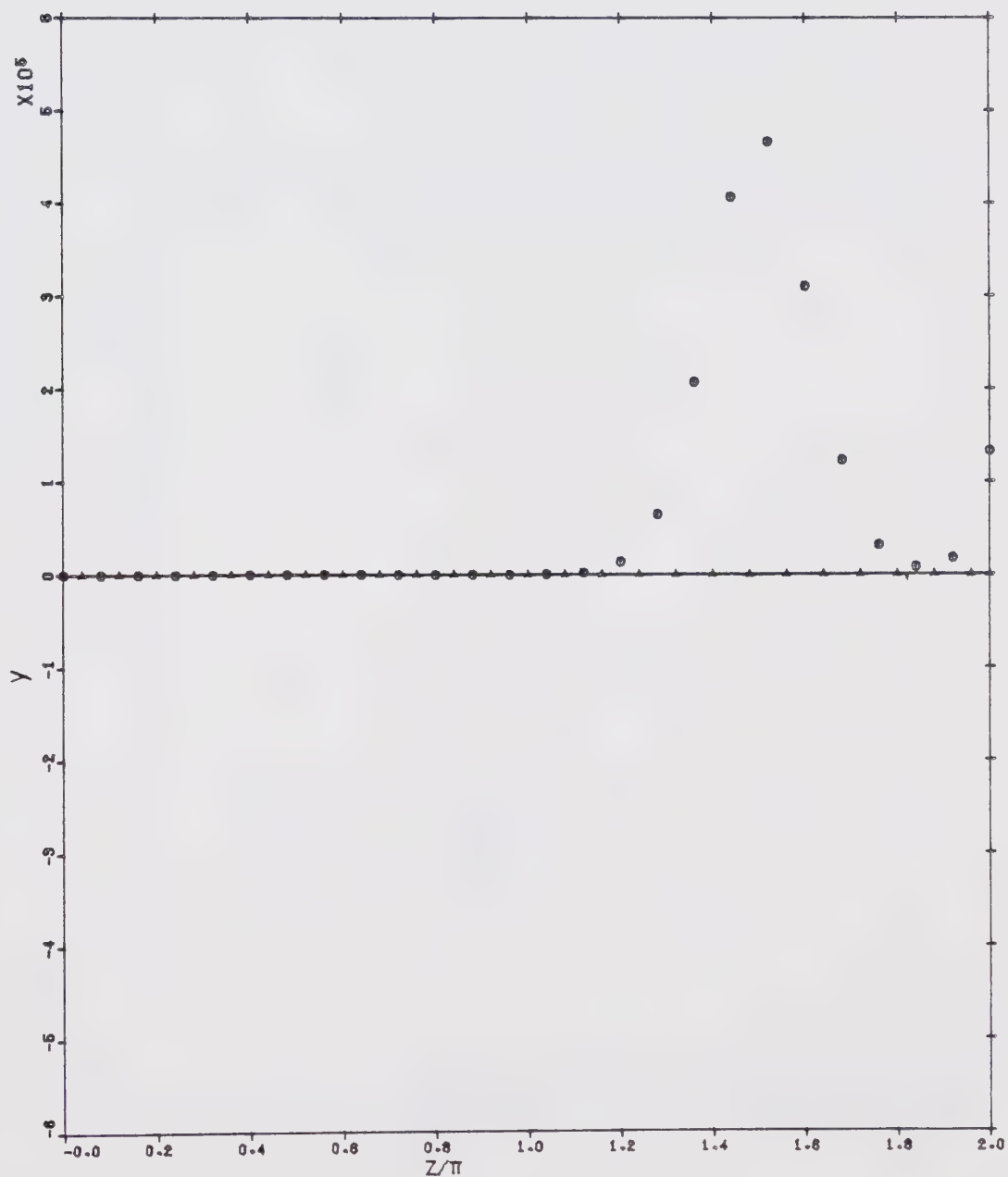
FIGURE 7.1 - SOLUTIONS TO MATHIEU EQUATION FOR
a_{GUESS} = -50.0, q = 20.0, $\Delta z = \pi/200.00$,
y(0) = 0.0, y'(0) = 1.0

(N.B.: Runge-Kutta solution uses [5].)

CHAR. NUM. (EST. VALUE) = -50.0000
 CHAR. NUM. (FINAL VALUE) = -31.3134
 $q = 20.0000$ $\Delta z = \pi/200.0$

$y(0) = 0.0000$ $y'(0) = 1.0000$

— SERIES APPROXIMATION
 ○ PIECEWISE LINEARIZATION
 △ RUNGE-KUTTA



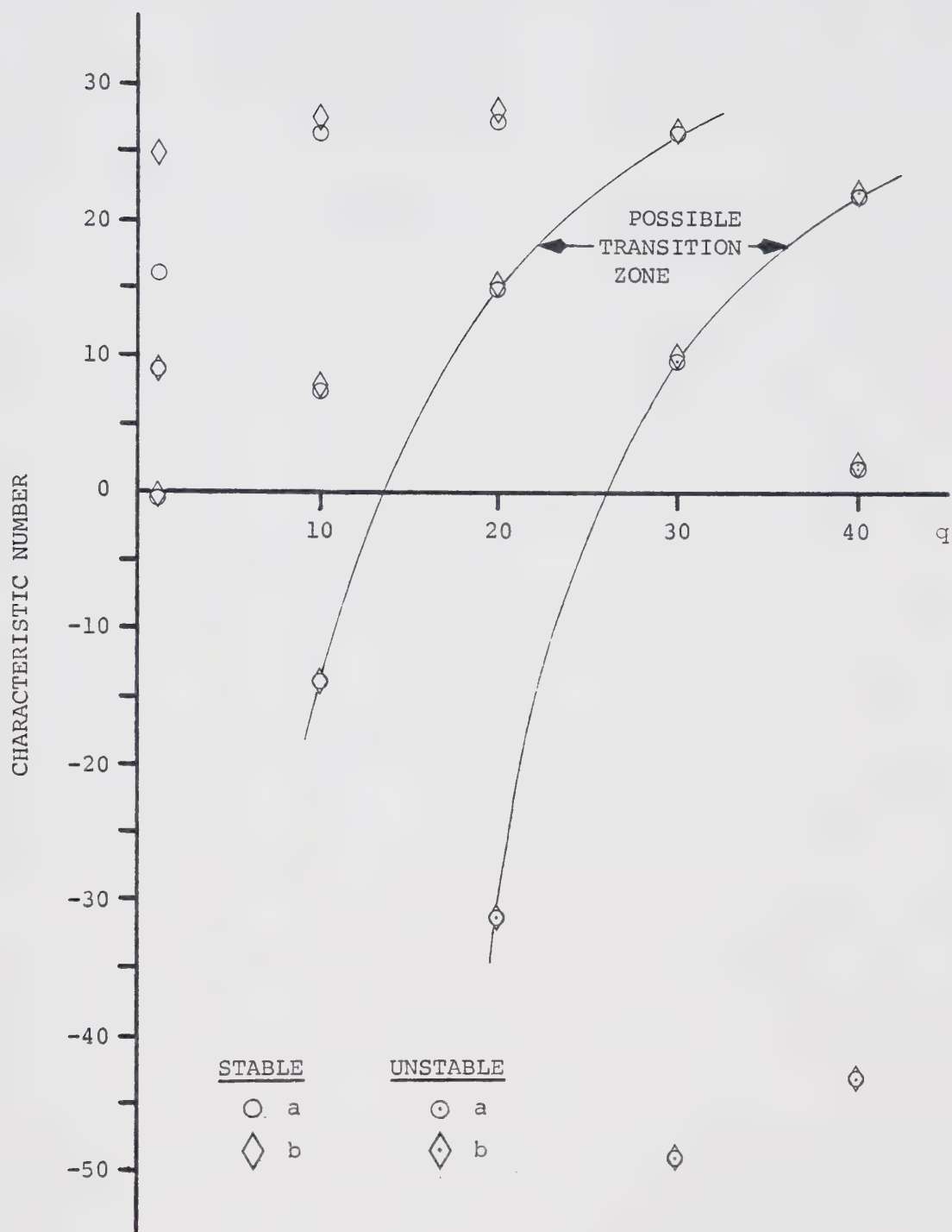


FIGURE 7.2 - STABILITY CHART FOR PIECEWISE LINEAR SOLUTION TO MATHIEU EQUATION

does indicate is that if this problem of the runaway piecewise linear solution cannot be resolved, this chart (or rather, a more accurate version of it) would give one an idea as to where the piecewise linear solution to the Mathieu equation becomes ineffective and another method should be pursued.

The timing results really posed no major problems as far as most of the values were concerned [27]. The piecewise linear method was fairly straightforward, and so, one can expect a linear solution, since as the number of increments is increased, the same calculations have to be done that many times more. The same applies to the series approximation to the exact solution, where the solution is mainly composed of the manipulation of the coefficient ratios.

The Runge-Kutta method [5] did give some unusual results. First, the magnitude of its times were suspect since one would expect the method to be faster than the others based on results for (3.1.1), (4.1.1), and (5.1.1). However, since the Mathieu equation is actually a linear one with a time-varying coefficient, it may be a difficult one for the Runge-Kutta routine [5] to manipulate. The piecewise linear method merely approximates the coefficient with a constant, and the series approximation to the exact solution is based upon the Mathieu functions.

The scatter in the timing results may have been a consequence of the size of the interval of z rather than the equation [60]. It appears that DVERK [5] itself may have been directly affected by the magnitude of Δz since (6.1.1) is a different type of equation than had been encountered in the previous chapters. How it is affected by it, and subsequent steps it executes could also explain why the execution times,

as determined by use of [27], were so large compared with those for the others.

VIII. CONCLUSIONS

1. The accuracy of the piecewise linear solution with respect to the one being compared will vary with the equation being examined. This is dependent upon the presence of damping, the degree of nonlinearity (if any) and whether the coefficients in these equations are constant or variable.

2. For those cases in which correspondence between the piecewise linear solution and the solution being compared is poor, there appear to be areas for which the piecewise linear method is unstable, or at least limited. The degree to which this occurs will depend upon the equation parameters and constants, as well as the size of the increments chosen, whether displacement or time.

3. The type of instabilities displayed by the piecewise linear solutions may be equation-dependent. That is, the instability seen in the Van der Pol equation appeared to be different than that displayed by the Mathieu equation. This may have to be confirmed by further work with these two equations plus the investigation of others.

4. For large increments of time or displacement (depending upon which equation was being solved), the piecewise linear method can become uneconomical since it will have a larger execution time compared with a smaller increment, and quite often can exceed the time limit set for the program used. However, under certain circumstances, the execution time of the piecewise linear method can be close to that of the chosen standard solution (such as the Runge-Kutta method [5] used as a comparison for the Van der Pol equation), or even cheaper (as was seen for the undamped hard spring and Mathieu equation).

5. Generally, the piecewise linearization is slower than the solution being compared for the system being examined except in the case of the undamped hard spring and the Mathieu equations. For the former, it was because the approximation to the complete elliptic integral of the first kind [15] had to be calculated each time the number of segments was increased, while for the Mathieu equation it was because the piecewise linearization was simpler than the approximation to the exact solution and the Runge-Kutta method [5]. It can be seen here, then, that the method of approximating the equation will determine the speed (and hence, the cost) of a run.

6. The tangential method of linearization is slower than that using chords, because of the various calculations that have to be carried out. The correspondence with the approximation to the exact solution [4], [15], [17] appears to be better for chords than for tangents, though the latter is more accurate when it comes to calculating period for the undamped hard spring equation.

IX. RECOMMENDATIONS FOR FURTHER WORK

Several questions arising from the results obtained still need to be answered. At the same time, there are some areas that can be investigated in more detail, extending past what has already been examined. The following, then, are some suggested subjects for further study.

A. Mathieu's Equation

The first thing that comes to mind is the situation with the instability of the piecewise linear method. The exact nature as to what was occurring has to be resolved, since it is unusual that the piecewise linearization solution had behaved rather well until it apparently broke down. The actual mechanism behind this has to be examined in depth to determine not only what is taking place, but also why, and if there is any means of rectifying this.

On the other hand, suppose that the reason for the instability observed in Figures 6.7, 6.9, 6.11, 6.13, 6.14, and 7.1 cannot be resolved. One alternative is to try and pursue other methods of solving this equation, using a different means of approximating the linear segment. If, however, all attempts to solve this problem fail, the only other thing that can be suggested is to obtain more data and fill in the gaps in Figure 7.2, to see if a boundary between stability and instability can be clearly defined.

In McLachlan [67], reference was made to an earlier paper of his [68] that dealt with fractional orders for the solutions. This reference, in turn, referred to a paper by Ince on the same subject [69].

In [68], a method for determining the order of the solution for a

particular a_{GUESS} and q is given, based on [69]. This may make the solution of (6.1.1) more general in nature compared with what had been done in Chapter VI, as fractional orders can now be included.

B. Two-Degree-of-Freedom Nonlinear Equations

Since many physical phenomena can be described by two-degree-of-freedom nonlinear equations, one possible approximate method of solution is piecewise linearization. However, the major difficulty is in the separation of the coupled equations that describe the situation into separate modes.

Some work on this separation has already been done in this area by Rosenberg [70], [71], which deals with specific classes of two-degree-of-freedom nonlinear differential equations. He also produced a major work dealing with this subject, and it appears that this may be of a more general nature [72].

Since this subject was merely touched upon, some work would be required in seeing what is needed for solving specific cases. Once the equations have been uncoupled (if such is possible for the particular situation being examined), the piecewise linear method can then be applied. These references may provide a starting point.

REFERENCES

1. Cunningham, W. J. Introduction to Nonlinear Analysis. New York, McGraw-Hill Book Company, 1958, p. 135-140.
2. Ibid., p. 133-135.
3. Ibid., p. 123-133.
4. Madderom, P. "JCBYF" in UBC Function--Mathematical (or Special) Functions. Vancouver, B. C., University of British Columbia Computing Centre, June, 1979, p. 40-41.
5. "DVERK", in IMSL Library Reference Manual, Seventh Edition, Vol. 1. Houston, Tex., IMSL, Inc., Jan., 1979, DVERK1-DVERK9.
6. Cunningham, op. cit., p. 66.
7. This is based on a diagram found on p.25, ch. 2 of the course notes for ELEC 557 given from Sept., 1979 to Apr., 1980 by Dr. A. C. Soudack, Professor, Dept. of Electrical Engineering, University of British Columbia, Vancouver, B. C.
8. Cunningham, op. cit., p. 75.
9. McLachlan, N. W. Ordinary Non-Linear Differential Equations In Engineering and Physical Sciences. London, Oxford University Press, 1956, p. 24-27.
10. Cunningham, op. cit., p. 74.
11. Soudack, op. cit., p. 5-63 - 5-86.
12. Cunningham, op. cit., p. 77-78.
13. Soudack, op. cit., p. 5-84 - 5-85.
14. Milne-Thomson, L. M. "Elliptic Functions", in Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, ed. M. Abramowitz and I. G. Stegun. New York, Dover

- Publications, Inc., 1972, p. 590.
15. Ibid., p. 591.
 16. Milne-Thomson, L. M. "Jacobian Elliptic Functions and Theta Functions", in Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, ed. M. Abramowitz and I. G. Stegun. New York, Dover Publications, Inc., 1972, p. 571.
 17. McLachlan, op. cit., p. 27.
 18. Timoshenko, S., et. al. Vibration Problems in Engineering, Fourth Edition. New York, John Wiley & Sons. Inc., 1974, p. 190-193.
 19. Ibid., p. 191.
 20. Riddle, Douglas F. Calculus and Analytic Geometry. Belmont, Calif., Wadsworth Publishing Co., Inc., 1970, p. 6.
 21. Timoshenko, et. al., op. cit., p. 192.
 22. Franklin, Phillip. "Algebra", in Mark's Standard Handbook for Mechanical Engineering, Eighth Edition, ed. Theodore Baumeister, et. al. New York, McGraw-Hill Book Company, 1978, p. 2-19.
 23. C R C Standard Mathematical Tables, 25th Edition, ed. William H. Beyer. Boca Raton, Florida, C R C Press, Inc., 1979, p. 170.
 24. Riddle, op. cit., p. 21, 22.
 25. University of Alberta Computing Services Overview 0120-0480, Computing Services. Edmonton, Alta., April 1980, p. 8.
 26. University of Alberta Computing Services Bulletin, Edmonton, Alta., N1511.0681, June 5, 1981, p. 1.
 27. University of Alberta Computing Services subroutine TIME. Its use was suggested by fellow graduate student, Roger Toogood, Dept. of Mechanical Engineering.
 28. Based on a diagram in Soudack, op. cit., p. 48, ch. 3.

29. Timoshenko, et. al., op. cit., p. 63-66.
30. Greenspan, Donald. Discrete Numerical Methods in Physics and Engineering. New York, Academic Press, Inc., 1974, p. 15.
31. This time limit was set by the author through his on-campus terminal which was connected to the University of Alberta Computing Services facilities.
32. Soudack, op. cit., p. 69, 70, ch. 3.
33. Timoshenko, et. al., op. cit., p. 64.
34. Timoshenko, et. al., op. cit., p. 63-71.
35. Timoshenko, et. al., op. cit., p. 66.
36. Timoshenko, et. al., op. cit., p. 71.
37. Timoshenko, et. al., op. cit., p. 69.
38. Based on diagrams in Soudack, op. cit., p. 72, ch. 3.
39. Based on diagrams in Soudack, op. cit., p. 71, ch. 3.
40. University of Alberta Computing Services Reference R131.0280, CGPL/CGPL2. Edmonton, Alta., February 1980, p. 1-12.
41. McLachlan, N. W. Theory and Application of Mathieu Functions. Oxford, Clarendon Press, 1947, p. 10.
42. Ince, E. L. Tables of the Elliptic Cylinder Functions, Proc. Roy. Soc. Edinburgh, Vol. 52, 1932, p. 355-423.
43. McLachlan, Theory and Application of Mathieu Functions, p. 10-41.
44. Ibid., p. 18.
45. Ibid., p. 21.
46. Ibid., p. 28.
47. Ince, op. cit., p. 358.
48. Ince, op. cit., p. 359.
49. Ince, op. cit., p. 360.

50. Ince, op. cit., p. 363.
51. McLachlan, Theory and Application of Mathieu Functions, p. 37.
52. Ibid., p. 29-31.
53. Ibid., p. 31.
54. Ibid., p. 30.
55. Ibid., p. 37.
56. Ibid., p. 25.
57. Ince, op. cit. p. 364-375.
58. Timoshenko, et. al., op. cit., p. 3, 4.
59. Program MA1-11A, "Differential Equations", Hewlett-Packard HP-67/HP-97 Math Pac 1, Corvallis, Oregon, 1976, p. 11-01 - 11-04, L11-01 - L11-02.
60. This information was obtained through consultation with Dr. R. Torgerson, Senior Analyst, University of Alberta Computing Centre, Edmonton, Alta.
61. Greenspan, op. cit., p. 52.
62. Fröberg, Carl-Erik. Introduction to Numerical Analysis, Second Edition. Reading, Mass., Addison-Wesley Publishing Co., Inc., 1969, p. 277.
63. Dorn, William S. and McCracken, Daniel D. Numerical Methods with Fortran IV Case Studies. New York, John Wiley and Sons, Inc., 1972, p. 389.
64. Ibid., p. 378-379.
65. Mayers, D. F. "Stability of Step-by-Step Methods", in Numerical Solutions of Ordinary and Partial Differential Equations, ed. L. Fox. Reading, Mass., Addison-Wesley Publishing Co., Inc., 1962, p. 48.

66. Ibid., p. 51.
67. McLachlan, Theory and Application of Mathieu Functions, p. 79, 378.
68. McLachlan, N. W. "Computation of the Solutions of Mathieu's Equation", Philosophical Magazine, Vol. 36, Series 7, 1945, p. 403-414.
69. Ince, E. L. "Mathieu Functions of Stable Type", Philosophical Magazine, Vol. 6, Series 7, 1928, p. 547-558.
70. Rosenberg, R. M. "On Normal Vibrations of a General Class of Nonlinear Dual-Mode Systems", Journal of Applied Mechanics, Transactions of the ASME, Vol. 83, Series E, June, 1961, p. 275-283.
71. Rosenberg, R. M. "The Normal Modes of Nonlinear n-Degree-of-Freedom Systems", Journal of Applied Mechanics, Transactions of the ASME, Vol. 84, Series E, March, 1962, p. 7-14.
72. Rosenberg, R. M. "On Nonlinear Vibrations of Systems with Many Degrees of Freedom", Advances in Applied Mechanics, Vol. 9. New York, Academic Press, p. 155-242.
73. Buttuls, Peter and Dembo, Stevyn. University of Alberta Computing Services Tutorial T11.1277, Digital Plotting System. Edmonton, Alberta, University of Alberta Computing Services, December, 1977.
74. "Mathematical Notation - Special Functions", in McGraw-Hill Dictionary of Scientific and Technical Terms, Second Edition, ed. Daniel N. Lapedes. New York, McGraw-Hill Book Company, 1978, p. A9.
75. Cress, Paul, et. al. FORTTRAN IV with WATFOR and WATFIV. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1970.

APPENDIX

The following programs were used in obtaining the results for this thesis:

- A. HSPRING - Hard spring equation (undamped)
- B. HSPRINGC - Hard spring equation (linear damping)
- C. HSPRINGQ - Hard spring equation (nonlinear damping)
- D. VANDERPOL - Van der Pol equation
- E. MATHIEU - Mathieu equation.

These programs were developed and run on the facilities of the University of Alberta Computing Services [25], [26], with the plot routines used being [40], [73]. Material upon which these programs are based is found in:

HSPRING - [4], [8], [9], [10], [11], [12], [14], [15],
[16], [18], [20], [22], [23], [24];
HSPRINGC - [5], [18], [20], [29], [30];
HSPRINGQ - [5], [18], [20], [29], [30], [74];
VANDERPOL - [5], [30], [32], [34], [38], [39], [74];
MATHIEU - [5], [34], [42], [43], [58].

The signum function as defined in [74] was used in HSPRINGQ and VANDERPOL. The major reference for the program code was [75].


```

1 C=====
2 C*
3 C*   PROGRAM HSPRING
4 C*
5 C*
6 C*
7 C*   THE FOLLOWING PROGRAM CALCULATES AND COMPARES THE SOLUTIONS
8 C*   FOR THE EQUATION:
9 C*
10 C*    $X(\text{DOUBLE-DOT})+A=X+B*(X**3)=0$ 
11 C*
12 C*   OBTAINED BY PIECEWISE LINERIZATION (USING CHORDS AND
13 C*   TANGENTS) WITH THAT OF AN EXACT RESULT, AND PLOTTING ALL
14 C*   THREE SETS OF VALUES.
15 C*
16 C*
17 C*   THE RESULTS ARE FOUND FROM THE FOLLOWING APPROXIMATIONS:
18 C*
19 C*   EXACT:   $X=XOE=CN(LAMBDA,OMEGA*T)$ 
20 C*
21 C*   CHORDS:   $X=XOSC+(XOC-XOSC)*COS(PC*(T-TOC))$ 
22 C*              $+ (XDOTOC/PC)*SIN(PC*(T-TOC))$ 
23 C*
24 C*              $X(DOT)=-PC*(XOC-XOSC)*SIN(PC*(T-TOC))$ 
25 C*              $+ XDOTOC*COS(PC*(T-TOC))$ 
26 C*
27 C*   TANGENTS:  $X=XOST+(XOC1-XOST)*COS(PT*(T-TOT))$ 
28 C*              $+ (XDOTOT/PT)*SIN(PT*(T-TOT))$ 
29 C*
30 C*              $X(DOT)=-PT*(XOC1-XOST)*SIN(PT*(T-TOT))$ 
31 C*              $+ XDOTOT*COS(PT*(T-TOT))$ 
32 C*
33 C*   THE VARIOUS PARAMETERS ARE EXPLAINED AND CALCULATED IN THE
34 C*   PROGRAM.
35 C*
36 C=====
37 C
38 C
39 C
40 C
41 C
42 C
43 C
44 C
45 C
46 C
47 C   IMPLICIT REAL*8 (A-H,O-Z)
48 C   REAL*8 DELTAX(200),TAUC(200),TAUT(200)
49 C   REAL*8 J,K,LAMBDA,M
50 C   REAL*4 AMPLE(200),AMPLC(200),AMPLT(200),TE(200),TC(200),TT(200)
51 C   REAL*4 YC(200),TCP(200),YT(200),TTP(200)
52 C   REAL*4 XA(2),YA(2)
53 C   REAL*4 HA,H8,HC,VA,Y8,VC
54 C   INTEGER*4 ALPH(20)
55 C   COMMON/COEFF/A,B
56 C   COMMON/ELLIP/PI,LAMBDA,K
57 C   COMMON/ENDOC/XOC
58 C   COMMON/ENDIC/XIC
59 C   COMMON/CROSSC/XOSC
60 C   COMMON/VIBC/PC
61 C   COMMON/ENDOT/XOT
62 C   COMMON/ENDIT/XIT
63 C   COMMON/CROSSI/XOST
64 C   COMMON/SHARE/XOC2
65 C   COMMON/VIST/PT
66 C
67 C
68 C   THE FOLLOWING VALUES ARE READ FROM DATA FILE HSPRINGDATA:
69 C
70 C
71 C   A,B=AS IN THE EQUATION
72 C
73 C   DISPLO=INITIAL AMPLITUDE
74 C   VO=INITIAL VELOCITY
75 C   TO=INITIAL TIME
76 C
77 C   HA,HC=PLOT PARAMETERS FOR HORIZONTAL AXIS (HC AN INTEGRAL MULTIPLE
78 C   OF 8.0)
79 C
80 C   VC=PLOT PARAMETER FOR VERTICAL AXIS (AN INTEGRAL MULTIPLE OF 7.0,
81 C   7.0 GIVING BEST RESULTS)
82 C
83 C   XA,YA=PARAMETERS FOR PLOTTING ZERO LINE
84 C
85 C   ALPH=ARRAY FOR PLOT AXIS LABELS
86 C
87 C=====
88 C   READ(5,10) A,B
89 C   10 FORMAT(2D10.6)
90 C   READ(5,20) DISPLO,VO,TO
91 C   20 FORMAT(3D10.6)
92 C   READ(5,30) HA,HC
93 C   30 FORMAT(2F10.4)
94 C   READ(5,40) VC
95 C   40 FORMAT(F10.4)
96 C   READ(5,50) XA(1)
97 C   50 FORMAT(F10.4)
98 C   READ(5,60) YA(1),YA(2)
99 C   60 FORMAT(2F10.4)
100 C   READ(5,70) (ALPH(I),I=1,12)
101 C   70 FORMAT(12A4)
102 C   READ(5,80) (ALPH(I),I=13,16)
103 C   80 FORMAT(4A4)
104 C   READ(5,90) (ALPH(I),I=17,20)
105 C   90 FORMAT(4A4)
106 C
107 C
108 C   PI=3.141592653589793
109 C   BETA=B*(DISPLO**2)/A
110 C=====

```



```

111 C
112 C   XOE=AMPLITUDE OF EXACT SOLUTION
113 C
114 C   LAMBDA=MODULUS (FOR EXACT SOLUTION)
115 C
116 C *****
117 XOE=DISPLO
118 LAMBDA=DSORT(0.5/((1.0/BETA)+1.0))
119 C *****
120 C
121 C   ELLINT=SUBROUTINE FOR CALCULATING THE COMPLETE ELLIPTIC INTEGRAL
122 C       OF THE FIRST KIND
123 C
124 C *****
125 CALL ELLINT
126 C *****
127 C
128 C   OMEGAE=EXACT SOLUTION ANGULAR FREQUENCY
129 C   TIME=CUMULATIVE TIME FOR EXACT SOLUTION
130 C   DELTAT=TIME INCREMENT
131 C
132 C *****
133 OMEGAE=DSORT(A*(1.0+BETA))
134 TIME=TO
135 DELTAT=K/(100.0*DSORT(A*(1.0+BETA)))
136 DO 100 N=1,101
137 U=OMEGAE*TIME
138 C *****
139 C
140 C   DJCBYF=SUBROUTINE FOR EVALUATING JACOBIAN ELLIPTIC FUNCTIONS
141 C       (SEE UBC MANUAL "UBC FUNCTION" FOR DETAILS)
142 C
143 C *****
144 CALL DJCBYF(U,LAMBDA,DSN,DCN,DDN,IE,6)
145 C *****
146 C
147 C   AMPL=ARRAY FOR INSTANTANEOUS AMPLITUDE (FOR PLOTTING)
148 C   TE=ARRAY FOR TIME (FOR PLOTTING)
149 C
150 C *****
151 AMPL(N)=XOE*DCN
152 TE(N)=TIME
153 TIME=TIME+DELTAT
154 100 CONTINUE
155 C *****
156 C
157 C   TAU=EXACT PERIOD
158 C   SIG=LARGEST VALUE OF QUARTER-PERIOD (FOR PLOT SCALING)
159 C
160 C *****
161 TAU=4.0*K/DSORT(A*(1.0+BETA))
162 SIG=TAU/4.0
163 C *****
164 C
165 C   THE FOLLOWING DO-LOOP (ENDING AT STEP 280) CALCULATES THE
166 C   PIECEWISE LINEAR PERIODS (FOR BOTH METHODS) BASED ON THE
167 C   PARTICULAR NUMBER OF SEGMENTS CHOSEN.
168 C
169 C *****
170 C
171 C   NC=NUMBER OF CHORDS (INITIALLY SET AT 5)
172 C   NT=NUMBER OF TANGENTS (INITIALLY SET AT 6)
173 C
174 C   XOC,XOT=INITIAL DISPLACEMENT FOR EACH SUCCESSIVE SEGMENT FOR
175 C       CHORDS AND TANGENTS, RESPECTIVELY
176 C
177 C   XDOTOC,XDOTOT=INITIAL VELOCITY FOR EACH SUCCESSIVE SEGMENT, AS
178 C       ABOVE
179 C   TOC,TOT=INITIAL TIME, FOR CHORDS AND TANGENTS, RESPECTIVELY
180 C
181 C   XOC1=INITIAL CROSSING POINT FOR TANGENT LINES OF TWO ADJACENT
182 C       SEGMENTS (SEE SUBROUTINE XSHARE BELOW)
183 C
184 C   DELTA=LENGTH OF SEGMENT
185 C   DELTAX=ARRAY FOR DELTA
186 C
187 C *****
188 NC=5
189 DO 280 I=1,20
190 NT=NC+1
191 H=NC
192 XOC=DISPLO
193 XOT=DISPLO
194 XDOTOC=VO
195 XDOTOT=VO
196 TOC=TO
197 TOT=TO
198 XOC1=DISPLO
199 DELTA=DISPLO/H
200 DELTAX(I)=DELTA
201 C *****
202 C
203 C   THE FOLLOWING SECTION CALCULATES THE PERIOD FOR THE METHOD OF
204 C   CHORDS (ENDING AT STEP 110).
205 C
206 C *****
207 C
208 C   X1C=DISPLACEMENT AT END OF SEGMENT
209 C
210 C *****
211 X1C=XOC-DELTA
212 DO 110 N=1,NC
213 C *****
214 C
215 C   XPC=SUBROUTINE FOR CALCULATING POINT WHERE SEGMENT CHORD LINE
216 C       CROSSES X-AXIS
217 C
218 C   FREQC=SUBROUTINE FOR CALCULATING ANGULAR FREQUENCY
219 C
220 C *****

```



```

221      CALL XPC
222      CALL FREOC
223      C*****
224      E
225      C      THE FOLLOWING SOLVES THE TRIGONOMETRIC EQUATION:
226      C
227      C       $C3C=C1C\cos(\text{THETAC})+C2C\sin(\text{THETAC})$ 
228      C
229      C      WHERE:
230      C
231      C       $\text{THETAC}=\text{PC}(\text{TIC}-\text{TOC})$ 
232      C
233      C      TIC BEING THE CUMULATIVE TIME TAKEN TO REACH THE END OF THE
234      C      SEGMENT BEING EXAMINED.
235      C
236      C*****
237      C       $C1C=XOC-XOSC$ 
238      C       $C2C=XDOTOC/PC$ 
239      C       $C3C=X1C-XOSC$ 
240      C       $\text{THETAC}=\text{DATAN}(C2C/C1C)+\text{DARCOS}(C3C/\text{DSORT}(C1C^2+C2C^2))$ 
241      C       $\text{TIC}=\text{TOC}+\text{THETAC}/PC$ 
242      C*****
243      E
244      C      AMPLC=ARRAY FOR XOC (FOR PLOTTING)
245      C      TC=ARRAY FOR TOC (FOR PLOTTING)
246      E
247      C*****
248      C      AMPLC(N)=XOC
249      C      TC(N)=TOC
250      C*****
251      E
252      C      RESETTING THE INITIAL CONDITIONS FOR THE NEXT SEGMENT.
253      E
254      C*****
255      C      XOC=X1C
256      C      X1C=XOC-DELTA
257      C      XDOTOC=-PC*(C1C*DSIN(THETAC)-C2C*DCOS(THETAC))
258      C      TOC=TIC
259      C      110 CONTINUE
260      C*****
261      E
262      C      TAUC=ARRAY FOR PERIOD FOUND USING CHORDS
263      E
264      C*****
265      C      TAUC(I)=4.0*TIC
266      C      IF(TIC.GE.BIG) BIG=TIC
267      C      NDC=NC+1
268      C      AMPLC(NDC)=XOC
269      C      TC(NDC)=TOC
270      C*****
271      E
272      C
273      C      THE FOLLOWING SECTION CALCULATES THE PERIOD FOR THE METHOD OF
274      C      TANGENTS (ENDING AT STEP 120).
275      E
276      E
277      C      X1T=DISPLACEMENT AT END OF SEGMENT
278      E
279      E
280      C*****
281      C      X1T=XOT-DELTA
282      C      DO 120 N=1,NT
283      C*****
284      E
285      C      XSHARE=SUBROUTINE FOR CALCULATING THE CROSSING POINT FOR TANGENT
286      C      LINES OF TWO ADJACENT SEGMENTS
287      E
288      C      XPT=SUBROUTINE FOR CALCULATING POINT WHERE SEGMENT TANGENT LINE
289      C      CROSSES X-AXIS
290      E
291      C      FREOT=SUBROUTINE FOR CALCULATING ANGULAR FREQUENCY
292      E
293      C*****
294      C      CALL XSHARE
295      C      CALL XPT
296      C      CALL FREOT
297      C*****
298      E
299      C      THE FOLLOWING SOLVES THE TRIGONOMETRIC EQUATION:
300      C
301      C       $C3T=C1T\cos(\text{THETAT})+C2T\sin(\text{THETAT})$ 
302      C
303      C      WHERE:
304      C
305      C       $\text{THETAT}=\text{PT}(\text{T1T}-\text{TOT})$ 
306      C
307      C      T1T BEING THE CUMULATIVE TIME TAKEN TO REACH THE END OF THE
308      C      SEGMENT BEING EXAMINED.
309      C
310      C*****
311      C       $C1T=XOC1-XOST$ 
312      C       $C2T=XDOTOT/PT$ 
313      C       $C3T=XOC2-XOST$ 
314      C      IF(DABS(XOT).LT.1.0D-10) C3T=-XOST
315      C       $\text{THETAT}=\text{DATAN}(C2T/C1T)+\text{DARCOS}(C3T/\text{DSORT}(C1T^2+C2T^2))$ 
316      C       $\text{T1T}=\text{TOT}+\text{THETAT}/PT$ 
317      C*****
318      E
319      C      AMPLT=ARRAY FOR XOC1 (FOR PLOTTING)
320      C      TT=ARRAY FOR TOT (FOR PLOTTING)
321      E
322      C*****
323      C      AMPLT(N)=XOC1
324      C      TT(N)=TOT
325      C*****
326      E
327      C      RESETTING THE INITIAL CONDITIONS FOR THE NEXT SEGMENT.
328      E
329      C*****
330      C      XOT=X1T

```



```

331      X1T=XOT-DELTA
332      XD0TOT=-PT*(C1T*DSIN(THETAT)-C2T*DCDS(THETAT))
333      TOT=T1T
334      XOC1=XOC2
335      120 CONTINUE
336 C #####
337 C
338 C      TAUT=PERIOD FOUND USING TANGENTS
339 C
340 C #####
341 C      TAUT(1)=8.0=T1T
342 C      IF(T1T.GE.BIG) BIG=T1T
343 C      NDT=NT+1
344 C      AMPLT(NDT)=XOC1
345 C      TT(NDT)=TOT
346 C #####
347 C
348 C      THE NEXT FEW LINES ARE USED IN FINALIZING ALL THE PLOT PARAMETERS
349 C      WHICH INCLUDES AXIS SCALING.
350 C
351 C #####
352 C      M=1
353 C      IF(DISPL0.LE.1.0) GO TO 140
354 C      130 CONTINUE
355 C      DECD=DISPL0/(10.0==M)
356 C      IF(DECD.LE.1.0) GO TO 160
357 C      M=M+1
358 C      GO TO 130
359 C      140 CONTINUE
360 C      DIGD=DISPL0*(10.0==M)
361 C      IF(DIGD.GE.1.0) GO TO 150
362 C      M=M+1
363 C      GO TO 140
364 C      150 CONTINUE
365 C      DECD=DIGD/10.0
366 C      M=(M-1)
367 C      160 CONTINUE
368 C      IF(DECD.LE.0.1) VFACT=0.1
369 C      IF((DECD.GT.0.1).AND.(DECD.LE.0.25)) VFACT=0.25
370 C      IF((DECD.GT.0.25).AND.(DECD.LE.0.5)) VFACT=0.5
371 C      IF((DECD.GT.0.5).AND.(DECD.LE.1.0)) VFACT=1.0
372 C      VB=(VFACT/(VC-2.0))=10.0==M
373 C      VA=-VB
374 C
375 C      M=1
376 C      IF(BIG.LE.1.0) GO TO 180
377 C      170 CONTINUE
378 C      DECT=BIG/(10.0==M)
379 C      IF(DECT.LE.1.0) GO TO 200
380 C      M=M+1
381 C      GO TO 170
382 C      180 CONTINUE
383 C      DIGT=BIG*(10.0==M)
384 C      IF(DIGT.GE.1.0) GO TO 190
385 C      M=M+1
386 C      GO TO 180
387 C      190 CONTINUE
388 C      DECT=DIGT/10.0
389 C      M=(M-1)
390 C      200 CONTINUE
391 C      IF(DECT.LE.0.08) HFACT=0.1
392 C      IF((DECT.GT.0.08).AND.(DECT.LE.0.16)) HFACT=0.2
393 C      IF((DECT.GT.0.16).AND.(DECT.LE.0.4)) HFACT=0.5
394 C      IF((DECT.GT.0.4).AND.(DECT.LE.0.8)) HFACT=1.0
395 C      IF((DECT.GT.0.8).AND.(DECT.LE.1.0)) HFACT=2.0
396 C      HB=HFACT*10.0==(M-1)
397 C      LONG=HC
398 C      XA(2)=DFLOAT(LONG)=HB-0.005
399 C #####
400 C
401 C      THE FOLLOWING SECTION SETS UP THE CHORD AND TANGENT RESULTS SO
402 C      THAT 8 POINTS OF THE CHORD SOLUTION AND 7 OF THE TANGENT VALUES
403 C      ARE PLOTTED FOR A CLEARER PRESENTATION, WITH THE PLOTS BEING
404 C      ALTERNATED FOR COMPARISON PURPOSES.
405 C
406 C      YC=ARRAY FOR PLOTTED CHORD DISPLACEMENTS
407 C      TCP=ARRAY FOR PLOTTED CHORD TIME VALUES
408 C      YT=ARRAY FOR PLOTTED TANGENT DISPLACEMENTS
409 C      TTP=ARRAY FOR PLOTTED TANGENT TIME VALUES
410 C
411 C #####
412 C      INCC=NDC/5
413 C      INCT=NDT/5
414 C      INCRT=INCT/2
415 C      IF(NDT.LT.10) INCRT=1
416 C      NCC=1
417 C      NCT=1
418 C      NPC=1
419 C      NPT=1
420 C      210 CONTINUE
421 C      YC(NPC)=AMPLC(NCC)
422 C      TCP(NPC)=TC(NCC)
423 C      IF(NCC.EQ.NDC) GO TO 220
424 C      NPC=NPC+1
425 C      NCC=NCC+INCC
426 C      GO TO 210
427 C      220 CONTINUE
428 C      YT(NPT)=AMPLT(NCT)
429 C      TTP(NPT)=TT(NCT)
430 C      IF(NCT.EQ.1) GO TO 230
431 C      NCT=NCT+INCT
432 C      GO TO 240
433 C      230 CONTINUE
434 C      NCT=NCT+INCRT
435 C      240 CONTINUE
436 C      IF(NCT.GE.NDT) GO TO 250
437 C      NPT=NPT+1
438 C      GO TO 220
439 C      250 CONTINUE
440 C      NPT=NPT+1

```



```

441      YT(NPT)=AMPLT(NDT)
442      TTP(NPT)=TT(NDT)
443  C #####
444  C
445  C      PLOTTING THE SOLUTIONS.  CONSULT THE WRITEUP ON CGPL/CGPL2 AND
446  C      THE MANUAL ON DIGITAL PLOTTING FOR DETAILS.
447  C
448  C #####
449  C
450      ND=101
451      NF=1
452      CALL CGPL2(TE,AMPLE,ND,NF,5,HA,HB,HC,VA,YB,VC,ALPH)
453      ND=NPC
454      CALL CGPL2(TCP,YC,ND,2,1,HA,HB,HC,VA,YB,VC,ALPH)
455      ND=NPT
456      CALL CGPL2(TTP,YT,ND,3,1,HA,HB,HC,VA,YB,VC,ALPH)
457      NF=4
458      CALL CGPL2(XA,YA,2,NF,4,HA,HB,HC,VA,YB,VC,ALPH)
459      HORIZ=HA
460      VERT=VC+0.5
461      CALL PLOT(HORIZ,VERT,3)
462      HORIZ=HC
463      CALL PLOT(HORIZ,VERT,2)
464      VERT=VERT+5.0
465      CALL PLOT(HORIZ,VERT,2)
466      HORIZ=HA
467      CALL PLOT(HORIZ,VERT,2)
468      VERT=VC+0.5
469      CALL PLOT(HORIZ,VERT,2)
470      STARTX=(HC-7.0)/2.0
471      STARTY=VC+0.5
472      HORIZ=STARTX+0.2
473      VERT=STARTY+4.5
474      CALL SYMBOL(HORIZ,VERT,0.2,'A =',0.0,3)
475      HORIZ=STARTX+1.0
476      CALL NUMBER(HORIZ,VERT,0.2,A,0.0,4)
477      HORIZ=STARTX+3.5
478      CALL SYMBOL(HORIZ,VERT,0.2,'B =',0.0,3)
479      HORIZ=STARTX+4.3
480      CALL NUMBER(HORIZ,VERT,0.2,B,0.0,4)
481      HORIZ=STARTX+0.2
482      VERT=STARTY+3.9
483      CALL SYMBOL(HORIZ,VERT,0.2,'NO. OF SEGMENTS (APPROX. TO EXACT',0.0
484      1,33)
485      HORIZ=STARTX+0.6
486      VERT=STARTY+3.6
487      CALL SYMBOL(HORIZ,VERT,0.2,'SOLN.') = 100',0.0,12)
488      HORIZ=STARTX+0.2
489      VERT=STARTY+3.2
490      CALL SYMBOL(HORIZ,VERT,0.2,'NO. OF CHORDS =',0.0,15)
491      HORIZ=STARTX+3.4
492      CPOINT=OFLOAT(HC)
493      CALL NUMBER(HORIZ,VERT,0.2,CPOINT,0.0,-1)
494      HORIZ=STARTX+0.2
495      VERT=STARTY+2.8
496      CALL SYMBOL(HORIZ,VERT,0.2,'NO. OF TANGENTS =',0.0,17)
497      HORIZ=STARTX+3.8
498      TPOINT=OFLOAT(NT)
499      CALL NUMBER(HORIZ,VERT,0.2,TPOINT,0.0,-1)
500      HORIZ=STARTX+0.2
501      VERT=STARTY+2.0
502      CALL SYMBOL(HORIZ,VERT,0.2,'X(O) =',0.0,6)
503      HORIZ=STARTX+1.6
504      CALL NUMBER(HORIZ,VERT,0.2,DISPO,0.0,4)
505      HORIZ=STARTX+3.5
506      CALL SYMBOL(HORIZ,VERT,0.2,'X(O) =',0.0,6)
507      HORIZ=STARTX+3.525
508      VERT=STARTY+2.3
509      CALL SYMBOL(HORIZ,VERT,0.2,75,0.0,-1)
510      HORIZ=STARTX+4.9
511      VERT=STARTY+2.0
512      CALL NUMBER(HORIZ,VERT,0.2,VO,0.0,4)
513      HORIZ=STARTX+0.3
514      VERT=STARTY+1.1
515      CALL PLOT(HORIZ,VERT,3)
516      HORIZ=STARTX+0.8
517      CALL PLOT(HORIZ,VERT,2)
518      HORIZ=STARTX+1.0
519      VERT=STARTY+1.0
520      CALL SYMBOL(HORIZ,VERT,0.2,'APPROX. TO EXACT SOLN.',0.0,22)
521      HORIZ=STARTX+0.6
522      VERT=STARTY+0.7
523      CALL SYMBOL(HORIZ,VERT,0.2,1,0.0,-1)
524      HORIZ=STARTX+1.0
525      VERT=STARTY+0.6
526      CALL SYMBOL(HORIZ,VERT,0.2,'CHORDS',0.0,6)
527      HORIZ=STARTX+0.6
528      VERT=STARTY+0.3
529      CALL SYMBOL(HORIZ,VERT,0.2,2,0.0,-1)
530      HORIZ=STARTX+1.0
531      VERT=STARTY+0.2
532      CALL SYMBOL(HORIZ,VERT,0.2,'TANGENTS',0.0,8)
533      NF=0
534      CALL CGPL2(XA,YA,2,NF,4,HA,HB,HC,VA,YB,VC,ALPH)
535  C #####
536  C
537  C      INCREMENTING THE NUMBER OF LINEAR SEGMENTS.
538  C
539  C #####
540  C
541      NC=NC+5
542      250 CONTINUE
543  C #####
544  C
545  C      THE CALCULATED VALUES ARE WRITTEN INTO OUTPUT FILE HSPRINGNUM
546  C
547  C #####
548  C
549      WRITE(10,270)
550      270 FORMAT('1',/PERIODS FOR THE EQUATION:')
551      WRITE(10,280)
552      280 FORMAT('0',/X(DOUBLE-DOT)+A*X+B=(X*=3)=0')
553      WRITE(10,290)

```



```

551      290 FORMAT('O','OBTAINED BY PIECEWISE LINEARIZATION')
552      WRITE(10,300) A,B
553      300 FORMAT('O','A=',G20.10,2X,'B=',G20.10)
554      WRITE(10,310) DISPL0,V0
555      310 FORMAT('O','X(0)=',G20.10,2X,'X(DOT)(0)=',G20.10)
556      WRITE(10,320) TAUE
557      320 FORMAT('O','EXACT SOLUTION PERIOD=',G20.10)
558      WRITE(10,330)
559      330 FORMAT(' ','DELTA X',16X,'CHORDS',15X,'TANGENTS')
560      DO 350 I=1,20
561      WRITE(10,340) DELTAX(I),TAUC(I),TAUT(I)
562      340 FORMAT(1X,3(G20.10,2X))
563      350 CONTINUE
564      STOP
565      END
566      C
567      C
568      C
569      C
570      C
571      C
572      C
573      C
574      C
575      C
576      C*****
577      C
578      C      THE FOLLOWING SUBROUTINE CALCULATES THE FIRST ONE HUNDRED TERMS OF
579      C      AN INFINITE SERIES APPROXIMATION OF THE COMPLETE ELLIPTIC INTEGRAL
580      C      OF THE FIRST KIND. SEE SECT. 17.3, P. 591, "HANDBOOK OF
581      C      MATHEMATICAL FUNCTIONS WITH FORMULAS, GRAPHS, AND MATHEMATICAL
582      C      TABLES", M. ABRAMOWITZ AND I. G. STEGUN, FOR DETAILS.
583      C
584      C      K=CALCULATED VALUE
585      C
586      C*****
587      C      SUBROUTINE ELLINT
588      C      IMPLICIT REAL*8 (A-H,O-Z)
589      C      REAL*8 J,K,LAMBDA,M
590      C      COMMON/ELLIP/PI,LAMBDA,K
591      C      M=LAMBDA**2
592      C      COEFF=1.0/2.0
593      C      SUM=0.25*M
594      C      DO 300 N=1,98
595      C      J=N+1.0
596      C      COEFF=COEFF*(2.0+J-1.0)/(2.0+J)
597      C      TERM=(COEFF**2)*(M**=(N+1))
598      C      SUM=SUM+TERM
599      300 CONTINUE
600      K=(PI/2.0)*(1.0+SUM)
601      RETURN
602      END
603      C
604      C
605      C
606      C*****
607      C
608      C      THE FOLLOWING SUBROUTINE CALCULATES THE POINT WHERE THE INTERVAL
609      C      CHORD LINE WOULD CROSS THE X-AXIS.
610      C
611      C      PC=CALCULATED VALUE
612      C      XOSC=CALCULATED VALUE
613      C
614      C*****
615      C      SUBROUTINE XPC
616      C      IMPLICIT REAL*8 (A-H,O-Z)
617      C      COMMON/COEFF/A,B
618      C      COMMON/ENDOC/XOC
619      C      COMMON/ENDIC/XIC
620      C      COMMON/CROSSC/XOSC
621      C      FO=A*XOC+B*(XOC**3)
622      C      F1=A*XIC+B*(XIC**3)
623      C      XOSC=(XIC-(F1/FO)*XOC)/(1-(F1/FO))
624      C      RETURN
625      C      END
626      C
627      C
628      C
629      C*****
630      C
631      C      THE FOLLOWING SUBROUTINE CALCULATES THE ANGULAR FREQUENCY FOR
632      C      AN INTERVAL USING CHORDS.
633      C
634      C*****
635      C      SUBROUTINE FREQC
636      C      IMPLICIT REAL*8 (A-H,O-Z)
637      C      COMMON/COEFF/A,B
638      C      COMMON/ENDOC/XOC
639      C      COMMON/ENDIC/XIC
640      C      COMMON/VIBC/PC
641      C      FO=A*XOC+B*(XOC**3)
642      C      F1=A*XIC+B*(XIC**3)
643      C      SLOPE=(FO-F1)/(XOC-XIC)
644      C      PC=DSQRT(SLOPE)
645      C      RETURN
646      C      END
647      C
648      C
649      C
650      C*****
651      C
652      C      THE FOLLOWING SUBROUTINE CALCULATES THE POINT WHERE THE TANGENT
653      C      LINES OF TWO ADJACENT INTERVALS WOULD CROSS.
654      C
655      C      XOC2=CALCULATED VALUE
656      C
657      C*****
658      C      SUBROUTINE XSHARE
659      C      IMPLICIT REAL*8 (A-H,O-Z)
660      C      COMMON/COEFF/A,B

```



```

661      COMMON/ENDOT/XOT
662      COMMON/ENDIT/XIT
663      COMMON/SHARE/XOC2
664      IF (DABS(XOT).LT.1.0D-10) GO TO 400
665      SLOPE0=A+3.0*B*(XOT**2)
666      SLOPE1=A+3.0*B*(XIT**2)
667      FO=A*XOT+B*(XOT**3)
668      F1=A*XIT+B*(XIT**3)
669      XOC2=(SLOPE0-XOT-SLOPE1*XIT-(FO-F1))/(SLOPE0-SLOPE1)
670      GO TO 410
671 400 CONTINUE
672      XOC2=X.00
673 410 CONTINUE
674      RETURN
675      ENH
676
677  E
678  E
679  E
680  E
681  E
682  E
683  E
684  E
685  E
686  E
687  E
688  E
689  E
690  E
691  E
692  E
693  E
694  E
695  E
696  E
697  E
698  E
699  E
700  E
701  E
702  E
703  E
704  E
705  E
706  E
707  E
708  E
709  E
710  E
711  E
712  E
713  E
714  E
715  E
716  E
717  E
718  E
719  E
720  E
721  E
722  E
723  E
724  E
725  E
726  E
727  E
728  E
729  E
730  E
731  E
732  E
733  E
734  E
735  E
736  E
737  E
738  E
739  E
740  E
741  E
742  E
743  E
744  E
745  E
746  E
747  E
748  E
749  E
750  E
751  E
752  E
753  E
754  E
755  E
756  E
757  E
758  E
759  E
760  E
761  E
762  E
763  E
764  E
765  E
766  E
767  E
768  E
769  E
770  E
771  E
772  E
773  E
774  E
775  E
776  E
777  E
778  E
779  E
780  E
781  E
782  E
783  E
784  E
785  E
786  E
787  E
788  E
789  E
790  E
791  E
792  E
793  E
794  E
795  E
796  E
797  E
798  E
799  E
800  E
801  E
802  E
803  E
804  E
805  E
806  E
807  E
808  E
809  E
810  E
811  E
812  E
813  E
814  E
815  E
816  E
817  E
818  E
819  E
820  E
821  E
822  E
823  E
824  E
825  E
826  E
827  E
828  E
829  E
830  E
831  E
832  E
833  E
834  E
835  E
836  E
837  E
838  E
839  E
840  E
841  E
842  E
843  E
844  E
845  E
846  E
847  E
848  E
849  E
850  E
851  E
852  E
853  E
854  E
855  E
856  E
857  E
858  E
859  E
860  E
861  E
862  E
863  E
864  E
865  E
866  E
867  E
868  E
869  E
870  E
871  E
872  E
873  E
874  E
875  E
876  E
877  E
878  E
879  E
880  E
881  E
882  E
883  E
884  E
885  E
886  E
887  E
888  E
889  E
890  E
891  E
892  E
893  E
894  E
895  E
896  E
897  E
898  E
899  E
900  E
901  E
902  E
903  E
904  E
905  E
906  E
907  E
908  E
909  E
910  E
911  E
912  E
913  E
914  E
915  E
916  E
917  E
918  E
919  E
920  E
921  E
922  E
923  E
924  E
925  E
926  E
927  E
928  E
929  E
930  E
931  E
932  E
933  E
934  E
935  E
936  E
937  E
938  E
939  E
940  E
941  E
942  E
943  E
944  E
945  E
946  E
947  E
948  E
949  E
950  E
951  E
952  E
953  E
954  E
955  E
956  E
957  E
958  E
959  E
960  E
961  E
962  E
963  E
964  E
965  E
966  E
967  E
968  E
969  E
970  E
971  E
972  E
973  E
974  E
975  E
976  E
977  E
978  E
979  E
980  E
981  E
982  E
983  E
984  E
985  E
986  E
987  E
988  E
989  E
990  E
991  E
992  E
993  E
994  E
995  E
996  E
997  E
998  E
999  E
1000  E

```

THE FOLLOWING SUBROUTINE CALCULATES THE POINT WHERE THE INTERVAL
TANGENT LINE WOULD CROSS THE X-AXIS.

XOST=CALCULATED VALUE

SUBROUTINE XPT
IMPLICIT REAL*8 (A-H,O-Z)
COMMON/COEFF/A,B
COMMON/ENDOT/XOT
COMMON/ENDIT/XIT
COMMON/CROSS/XOST
F=A*XOT+B*(XOT**3)
SLOPE=A+3.0*B*(XOT**2)
XOST=XOT-F/SLOPE
RETURN
END

THE FOLLOWING SUBROUTINE CALCULATES THE ANGULAR FREQUENCY FOR
AN INTERVAL USING TANGENTS.

PT=CALCULATED VALUE

SUBROUTINE FROOT
IMPLICIT REAL*8 (A-H,O-Z)
COMMON/COEFF/A,B
COMMON/ENDOT/XOT
COMMON/VIBT/PT
SLOPE=A+3.0*B*(XOT**2)
PT=DSQRT(SLOPE)
RETURN
END

END OF FILE


```

1  C=====
2  C
3  C= PROGRAM HSPRINGC
4  C
5  C=
6  C=
7  C= THE FOLLOWING PROGRAM CALCULATES THE SOLUTION FOR THE
8  C= EQUATION:
9  C=
10 C=  $X(\text{DOUBLE-DOT}) + C = X(\text{DOT}) + A = X + B = (X = 3) = 0$ 
11 C=
12 C= OBTAINED BY PIECEWISE LINEARIZATION USING CHORDS, AS WELL AS
13 C= FINDING THE FIRST TROUGH ON THE DISPLACEMENT-TIME CURVE.
14 C=
15 C=
16 C= THE RESULTS ARE FOUND FROM THE FOLLOWING:
17 C=
18 C=  $X = X_{\text{STAR}} + \exp(-N \cdot (T - T_0)) \cdot (C_1 \cdot \cos(P_{\text{STAR}} \cdot (T - T_0))$ 
19 C=  $+ C_2 \cdot \sin(P_{\text{STAR}} \cdot (T - T_0))$ 
20 C=
21 C=  $X(\text{DOT}) = \exp(-N \cdot (T - T_0)) \cdot (C_3 \cdot \cos(P_{\text{STAR}} \cdot (T - T_0))$ 
22 C=  $+ C_4 \cdot \sin(P_{\text{STAR}} \cdot (T - T_0))$ 
23 C=
24 C=  $C_1 = X_0 - X_{\text{STAR}}$ 
25 C=  $C_2 = (N \cdot (X_0 - X_{\text{STAR}}) + X_{\text{DOT}0}) / P$ 
26 C=  $C_3 = -N \cdot C_1 + P \cdot C_2$ 
27 C=  $C_4 = -P \cdot C_1 - N \cdot C_2$ 
28 C=
29 C= THE VARIOUS PARAMETERS ARE EXPLAINED AND CALCULATED IN THE
30 C= PROGRAM.
31 C=
32 C=
33 C= THE PROGRAM ALSO SOLVES THE EQUATION USING FIFTH- AND SIXTH-
34 C= ORDER RUNGE-KUTTA METHODS. BOTH SOLUTIONS ARE PLOTTED.
35 C=====
36 C
37 C
38 C
39 C
40 C
41 C
42 C
43 C
44 C
45 C
46 C
47 C IMPLICIT REAL*8 (A-H,O-Z)
48 C EXTERNAL FCN
49 C REAL*8 Y(2),CN(24),W(2,20)
50 C REAL*8 N
51 C REAL*4 AMPLP(200),AMPLRK(200),TP(200),TRK(200)
52 C REAL*4 YP(2000),TPP(200)
53 C REAL*4 XA(2),YA(2)
54 C REAL*4 HA,HB,HC,VA,VB,VC
55 C INTEGER*4 ALPH(20)
56 C COMMON/COEFF/A,B
57 C COMMON/DAMP/C
58 C
59 C
60 C THE FOLLOWING VALUES ARE READ FROM DATA FILE HSPRINGC.DAT:
61 C
62 C
63 C A,B,C=SAME AS ABOVE
64 C
65 C DISPL0=INITIAL AMPLITUDE
66 C V0=INITIAL VELOCITY
67 C TIME0=INITIAL TIME
68 C
69 C H=NUMBER OF DIVISIONS OF INITIAL DISPLACEMENT
70 C DTIME=TIME INCREMENT (RUNGE-KUTTA SOLUTION)
71 C
72 C HA,HC=PLOT PARAMETERS FOR HORIZONTAL AXIS (HC AN INTEGRAL MULTIPLE
73 C OF 7.0)
74 C
75 C VC=PLOT PARAMETER FOR VERTICAL AXIS (AN INTEGRAL MULTIPLE OF 12.0,
76 C 12.0 GIVING BEST RESULTS)
77 C
78 C XA,YA=PARAMETERS FOR PLOTTING ZERO LINE
79 C
80 C ALPH=ARRAY FOR PLOT AXIS LABELS
81 C
82 C
83 C
84 C READ(5,10) A,B,C
85 C 10 FORMAT(3D10.6)
86 C READ(5,20) DISPL0,V0,TIME0
87 C 20 FORMAT(3D10.6)
88 C READ(5,30) H,DTIME
89 C 30 FORMAT(2D10.6)
90 C READ(5,40) HA,HC
91 C 40 FORMAT(2F10.4)
92 C READ(5,50) VC
93 C 50 FORMAT(F10.4)
94 C READ(5,60) XA(1)
95 C 60 FORMAT(F10.4)
96 C READ(5,70) YA(1),YA(2)
97 C 70 FORMAT(2F10.4)
98 C READ(5,80) (ALPH(I),I=1,12)
99 C 80 FORMAT(12A4)
100 C READ(5,90) (ALPH(I),I=13,16)
101 C 90 FORMAT(4A4)
102 C READ(5,100) (ALPH(I),I=17,20)
103 C 100 FORMAT(4A4)
104 C
105 C N=DAMPING FACTOR
106 C
107 C N=C/2.0
108 C
109 C
110 C

```



```

111 C   SETTING THE INITIAL CONDITIONS.
112 C
113 C   XO=INITIAL DISPLACEMENT FOR EACH SUCCESSIVE SEGMENT
114 C   XDOTO=INITIAL VELOCITY FOR EACH SEGMENT
115 C   TO=CUMULATIVE TIME TAKEN TO REACH BEGINNING OF SEGMENT
116 C   DELT=INITIAL GUESS AT TIME INTERVAL FOR THE SEGMENT BEING EXAMINED
117 C   (SEE SUBROUTINE TIME BELOW)
118 C
119 C   DELTAX=LENGTH OF SEGMENT
120 C
121 C   I=COUNTER FOR DATA POINTS TO BE PLOTTED
122 C
123 C *****
124 C   XO=DISPLO
125 C   XDOTO=VO
126 C   TO=TIMEO
127 C   DELT=TO
128 C   DELTAX=DISPLO/H
129 C   DELX=DELTAX
130 C   I=0
131 C *****
132 C
133 C   THE EQUATION PARAMETERS ARE WRITTEN INTO OUTPUT FILE HSPRINGCNUM
134 C
135 C *****
136 C   WRITE(10,110)
137 C   110 FORMAT('1','PIECEWISE LINEAR PERIOD FOR THE EQUATION:')
138 C   WRITE(10,120)
139 C   120 FORMAT('0','X(DOUBLE-DOT)+C=X(DOT)+A=X+B=(X=3)=0')
140 C   WRITE(10,130) A,B,C
141 C   130 FORMAT('0','A=',G20.10,2X,'B=',G20.10,2X,'C=',G20.10)
142 C   WRITE(10,140) DISPLO,VO
143 C   140 FORMAT('0','X(0)=',G20.10,2X,'X(DOT)(0)=',G20.10)
144 C   WRITE(10,150)
145 C   150 FORMAT('1',6X,'AMPLITUDE',10X,'CUMULATIVE TIME')
146 C   160 CONTINUE
147 C   I=I+1
148 C *****
149 C
150 C   X1=DISPLACEMENT AT END OF SEGMENT
151 C
152 C   XPRIME=SUBROUTINE FOR CALCULATING POINT WHERE SEGMENT CHORD LINE
153 C   CROSSES X-AXIS
154 C   XSTAR=CALCULATED VALUE
155 C
156 C   FREQ=SUBROUTINE FOR CALCULATING ANGULAR FREQUENCY
157 C   PSTAR=CALCULATED VALUE
158 C
159 C   TIME=SUBROUTINE FOR CALCULATING TIME INTERVAL FOR A SEGMENT
160 C   DELTAT=CALCULATED VALUE
161 C
162 C   SPEED=SUBROUTINE FOR CALCULATING SPEED AT END OF SEGMENT
163 C   XDOT1=CALCULATED VALUE
164 C
165 C **NOTE**=THE PROGRAM STOPS IF THE DISPLACEMENT AT THE END OF A SEGMENT
166 C   IS EQUAL TO -DISPL (I. E., UNDAMPED CASE).
167 C
168 C *****
169 C   X1=XO-DELTAX
170 C   CALL XPRIME(XO,X1,XSTAR)
171 C   CALL FREQ(XO,X1,N,PSTAR)
172 C   CALL TIME(XO,XDOTO,X1,DELT,XSTAR,N,PSTAR,DELTAT)
173 C   CALL SPEED(XO,XDOTO,XSTAR,N,DELTAT,PSTAR,XDOT1)
174 C *****
175 C
176 C   STARTING AT STEP 190 AND CONTINUING AT 200, THE PROGRAM TESTS FOR
177 C   ZERO VELOCITY. IF IT OCCURS AT THE END OF THE SEGMENT, THE
178 C   PROGRAM STOPS (SEE STEP 300). IF NOT, THE LOCATION OF THE FIRST
179 C   TROUGH IS SOUGHT (STEPS 250 TO 270).
180 C
181 C *****
182 C   DELT=DELTAT
183 C   T1=TO+DELTAT
184 C   IF(XO.GE.O.O) GO TO 170
185 C   GO TO 190
186 C   170 CONTINUE
187 C *****
188 C
189 C   AMPLP=ARRAY FOR XO
190 C   TP=ARRAY FOR TO
191 C
192 C *****
193 C   AMPLP(I)=XO
194 C   TP(I)=TO
195 C *****
196 C
197 C   THE INTERIM VALUES ARE WRITTEN INTO OUTPUT FILE HSPRINGCNUM.
198 C
199 C *****
200 C   WRITE(10,180) XO,TO
201 C   180 FORMAT(1X,2(G20.10,2X))
202 C *****
203 C
204 C   RESETTING THE INITIAL CONDITIONS.
205 C
206 C *****
207 C   XO=X1
208 C   XDOTO=XDOT1
209 C   TO=T1
210 C   GO TO 180
211 C   190 CONTINUE
212 C   IF(DABS(XDOT1).LT.1.0D-08) GO TO 200
213 C   GO TO 170
214 C   200 CONTINUE
215 C *****
216 C
217 C   THE SEGMENT IN WHICH THE FIRST TROUGH OCCURS IS WRITTEN INTO
218 C   OUTPUT FILE HSPRINGCNUM.
219 C
220 C *****

```



```

221      AMPLP(I)=XO
222      TP(I)=TO
223      WRITE(10,210) XO,TO
224      210  FORMAT(1X,2(G20.10,2X))
225      DIFF=-DISPLO-XO
226      IF(DABS(DIFF).LT.1.00-08) GO TO 300
227      WRITE(10,220) X1,T1
228      220  FORMAT(1X,2(G20.10,2X))
229      WRITE(10,230)
230      230  FORMAT('-', 'TROUGH ITERATIONS')
231      WRITE(10,240)
232      240  FORMAT('0',10X,'X0',20X,'X1',14X,'X1(CALCULATED)')
233      C *****
234      C
235      C      THE FOLLOWING (ENDING AT STEP 270) IS THE CALCULATION OF THE FIRST
236      C      TROUGH. USING THE TIME INTERVAL REQUIRED TO REACH ZERO VELOCITY,
237      C      A NEW VALUE OF X1 (XF) IS CALCULATED AND COMPARED TO THE PREVIOUS
238      C      VALUE. THIS NEW VALUE BECOMES THE NEW X1 IF THE DIFFERENCE
239      C      BETWEEN XF AND THE OLD X1 IS NOT WITHIN A SET TOLERANCE, AND THE
240      C      PROCESS STARTS AGAIN.
241      C
242      C      THESE ITERATIONS ARE WRITTEN INTO OUTPUT FILE HSPRINGCNUM.
243      C
244      C *****
245      250  CONTINUE
246      C1=XO-XSTAR
247      C2=(N=(XO-XSTAR)+XDOTO)/PSTAR
248      P=PSTAR
249      XF=XSTAR+DEXP(-N*DELTAT)*(C1*DCOS(P*DELTAT)+C2*DSIN(P*DELTAT))
250      WRITE(10,260) XO,X1,XF
251      260  FORMAT(1X,3(G20.10,2X))
252      DIFF=X1-XF
253      IF(DABS(DIFF).LT.1.00-08) GO TO 270
254      X1=XF
255      DIFF=XO-XF
256      IF(DABS(DIFF).LT.1.00-08) GO TO 270
257      CALL XPRIME(XO,X1,XSTAR)
258      CALL FREQ(XO,X1,N,PSTAR)
259      CALL TIME(XO,XDOTO,X1,DELT,XSTAR,N,PSTAR,DELTAT)
260      GO TO 250
261      270  CONTINUE
262      C *****
263      C
264      C      THE FINAL VALUES ARE WRITTEN INTO OUTPUT FILE HSPRINGCNUM.
265      C
266      C *****
267      TF=T1
268      XF=X1
269      I=I+1
270      NPOINT=I
271      AMPLP(I)=XF
272      TP(I)=TF
273      TAU=2.0*T1
274      WRITE(10,280) DELTAX,TAU
275      280  FORMAT('-', 'DELTAX=',G20.10,2X,'PERIOD=',G20.10)
276      WRITE(10,290) XF
277      290  FORMAT('0', 'FIRST TROUGH OCCURS AT X=',G20.10)
278      GO TO 320
279      300  CONTINUE
280      C *****
281      C
282      C      THE FINAL VALUES ARE WRITTEN INTO OUTPUT FILE HSPRINGCNUM (UNDAMPED
283      C      CASE).
284      C
285      C *****
286      NPOINT=I
287      TAU=2.0*T1
288      WRITE(10,310) DELTAX,TAU
289      310  FORMAT('-', 'DELTAX=',G20.10,2X,'PERIOD=',G20.10)
290      320  CONTINUE
291      C *****
292      C
293      C      THE FOLLOWING SECTION (ENDING AT LINE 330) IS THE RUNGE-KUTTA
294      C      SOLUTION OF THE SAME ORIGINAL EQUATION.
295      C
296      C
297      C      TFINAL=FINAL TIME
298      C      INTVL=NUMBER OF INCREMENTS
299      C
300      C *****
301      TFINAL=T1+0.1
302      INTVL=TFINAL/DTIME+1
303      C *****
304      C
305      C      THE FOLLOWING ARE PARAMETERS FOR THE NUMERICAL SUBROUTINE.
306      C      CONSULT THE IMSL MANUAL ON "DVERK" FOR FURTHER DETAILS
307      C
308      C *****
309      XRK=TIMEO
310      Y(1)=DISPLO
311      Y(2)=VO
312      NEQN=2
313      NW=2
314      TOL=1.0D-12
315      IND=1
316      XEND=DTIME
317      C *****
318      C
319      C      ZEIT=CUMULATIVE VALUE OF TIME AT END OF EACH INCREMENT
320      C
321      C *****
322      ZEIT=0.0
323      C *****
324      C
325      C      AMPLRK=ARRAY FOR AMPLITUDE VALUES
326      C      TRK=ARRAY FOR TIME VALUES
327      C
328      C *****
329      DO 330 J=1,INTVL
330

```



```

331      AMPLRK(J)=Y(1)
332      TRK(J)=ZEIT
333      CALL DVERK1NEON,FCN,XRK,Y,XEND,TOL,IND,CN,NW,W,IER)
334      C*****
335      C
336      C      INCREMENTING THE TIME VALUES.
337      C
338      C*****
339      ZEIT=XEND
340      XEND=XEND+OTIME
341      330 CONTINUE
342      C*****
343      C
344      C      THE CALCULATED VALUES ARE WRITTEN INTO OUTPUT FILE HSPRINGCRK.
345      C
346      C*****
347      WRITE(11,340)
348      340 FORMAT('1','RUNGE-KUTTA RESULTS FOR THE EQUATION:')
349      WRITE(11,350)
350      350 FORMAT('0','X(DOUBLE-DOT)+C=X(DOT)+A=X+B=(X==3)=0')
351      WRITE(11,360) A,B,C
352      360 FORMAT('0','A=',G20.10,2X,'B=',G20.10,2X,'C=',G20.10)
353      WRITE(11,370) DISPLO,YO
354      370 FORMAT('0','X(0)=',G20.10,2X,'X(DOT)(0)=',G20.10)
355      WRITE(11,380)
356      380 FORMAT('0','9X,'TIME',15X,'AMPLITUDE')
357      DO 400 NO=1,INTVL
358      WRITE(11,390) TRK(NO),AMPLRK(NO)
359      390 FORMAT(1X,2(G20.10,2X))
360      400 CONTINUE
361      C*****
362      C
363      C      THE NEXT FEW LINES ARE USED IN FINALIZING ALL THE PLOT PARAMETERS
364      C      WHICH INCLUDES AXIS SCALING.
365      C
366      C*****
367      M=1
368      IF(DISPLO.LE.1.0) GO TO 420
369      410 CONTINUE
370      DECD=DISPLO/(10.0==M)
371      IF(DECD.LE.1.0) GO TO 440
372      M=M+1
373      GO TO 410
374      420 CONTINUE
375      DIGD=DISPLO/(10.0==M)
376      IF(DIGD.GE.1.0) GO TO 430
377      M=M+1
378      GO TO 420
379      430 CONTINUE
380      DECD=DIGD/10.0
381      M=(M-1)
382      440 CONTINUE
383      IF(DECD.LE.0.1) VFACT=0.1
384      IF((DECD.GT.0.1).AND.(DECD.LE.0.25)) VFACT=0.25
385      IF((DECD.GT.0.25).AND.(DECD.LE.0.5)) VFACT=0.5
386      IF((DECD.GT.0.5).AND.(DECD.LE.1.0)) VFACT=1.0
387      SFAC=VC/10.0
388      VA=-VFACT*SFAC*10.0==M
389      DEN=VC/(2.0-SFAC)
390      VB=(VFACT/DEN)*10.0==M
391      C
392      M=1
393      IF(TFINAL.LE.1.0) GO TO 480
394      450 CONTINUE
395      DECT=TFINAL/(10.0==M)
396      IF(DECT.LE.1.0) GO TO 480
397      M=M+1
398      GO TO 450
399      460 CONTINUE
400      DIGT=TFINAL/(10.0==M)
401      IF(DIGT.GE.1.0) GO TO 470
402      M=M+1
403      GO TO 460
404      470 CONTINUE
405      DECT=DIGT/10.0
406      M=(M-1)
407      480 CONTINUE
408      IF(DECT.LE.0.07) HFACT=0.1
409      IF((DECT.GT.0.07).AND.(DECT.LE.0.14)) HFACT=0.2
410      IF((DECT.GT.0.14).AND.(DECT.LE.0.35)) HFACT=0.5
411      IF((DECT.GT.0.35).AND.(DECT.LE.0.7)) HFACT=1.0
412      IF((DECT.GT.0.7).AND.(DECT.LE.1.0)) HFACT=2.0
413      HB=HFACT*10.0==(M-1)
414      LONG=HC
415      XA(2)=DFLOAT(LONG)=HB-0.005
416      C*****
417      C
418      C      THE FOLLOWING SECTION SETS UP THE PIECEWISE LINEAR RESULTS SO THAT
419      C      NO MORE THAN 13 POINTS ARE PLOTTED FOR A CLEARER PRESENTATION.
420      C
421      C      YP=ARRAY FOR PLOTTED PIECEWISE LINEAR DISPLACEMENTS
422      C      TPP=ARRAY FOR PLOTTED PIECEWISE LINEAR TIME VALUES
423      C
424      C*****
425      COUNT=DFLOAT(NPOINT)
426      INC=COUNT/10.0
427      DELINC=COUNT/10.0-DFLOAT(INC)
428      IF(DELINC.GE.0.5) INC=INC+1
429      NP=1
430      ND=1
431      490 CONTINUE
432      YP(NP)=AMPLP(ND)
433      TPP(NP)=TP(ND)
434      NP=NP+1
435      ND=ND+INC
436      IF(ND.GE.NPOINT) GO TO 500
437      GO TO 490
438      500 CONTINUE
439      YP(NP)=AMPLP(NPOINT)
440      TPP(NP)=TP(NPOINT)

```



```

441 C*****
442 C
443 C   PLOTTING THE SOLUTIONS.  CONSULT THE WRITEUP ON CGPL/CGPL2 AND
444 C   THE MANUAL ON DIGITAL PLOTTING FOR DETAILS.
445 C
446 C*****
447 ND=INTVL
448 NF=1
449 CALL CGPL2(TRK,AMPLRK,ND,NF,5,HA,HB,HC,VA,VB,VC,ALPH)
450 ND=NP
451 CALL CGPL2(TPP,YP,ND,2,1,HA,HB,HC,VA,VB,VC,ALPH)
452 NF=4
453 CALL CGPL2(XA,YA,2,NF,4,HA,HB,HC,VA,VB,VC,ALPH)
454 STARTX=(HC-6.3)/2.0
455 STARTY=VC+0.5
456 HORIZ=HA
457 VERT=VC+0.5
458 CALL PLOT(HORIZ,VERT,3)
459 HORIZ=HC
460 CALL PLOT(HORIZ,VERT,2)
461 VERT=VERT+4.6
462 CALL PLOT(HORIZ,VERT,2)
463 HORIZ=HA
464 CALL PLOT(HORIZ,VERT,2)
465 VERT=VC+0.5
466 CALL PLOT(HORIZ,VERT,2)
467 HORIZ=STARTX+0.2
468 VERT=STARTY+4.2
469 CALL SYMBOL(HORIZ,VERT,0.2,'A =',0.0,3)
470 HORIZ=STARTX+1.0
471 CALL NUMBER(HORIZ,VERT,0.2,A,0.0,4)
472 HORIZ=STARTX+3.0
473 CALL SYMBOL(HORIZ,VERT,0.2,'B =',0.0,3)
474 HORIZ=STARTX+3.8
475 CALL NUMBER(HORIZ,VERT,0.2,B,0.0,4)
476 HORIZ=STARTX+2.0
477 VERT=STARTY+3.8
478 CALL SYMBOL(HORIZ,VERT,0.2,'C =',0.0,3)
479 HORIZ=STARTX+2.8
480 CALL NUMBER(HORIZ,VERT,0.2,C,0.0,4)
481 HORIZ=STARTX+0.2
482 VERT=STARTY+3.1
483 CALL SYMBOL(HORIZ,VERT,0.2,52.0,0,-1)
484 HORIZ=STARTX+0.4
485 CALL SYMBOL(HORIZ,VERT,0.2,'X (PIECEWISE',0.0,11)
486 HORIZ=STARTX+1.0
487 VERT=STARTY+2.8
488 CALL SYMBOL(HORIZ,VERT,0.2,'LINEARIZATION) =',0.0,16)
489 HORIZ=STARTX+4.4
490 CALL NUMBER(HORIZ,VERT,0.2,DELX,0.0,4)
491 HORIZ=STARTX+0.2
492 VERT=STARTY+2.4
493 CALL SYMBOL(HORIZ,VERT,0.2,52.0,0,-1)
494 HORIZ=STARTX+0.4
495 CALL SYMBOL(HORIZ,VERT,0.2,'T (RUNGE-KUTTA) =',0.0,17)
496 HORIZ=STARTX+4.0
497 CALL NUMBER(HORIZ,VERT,0.2,DTIME,0.0,4)
498 HORIZ=STARTX+0.2
499 VERT=STARTY+1.6
500 CALL SYMBOL(HORIZ,VERT,0.2,'X(0) =',0.0,6)
501 HORIZ=STARTX+1.6
502 CALL NUMBER(HORIZ,VERT,0.2,DISPLO,0.0,4)
503 HORIZ=STARTX+3.5
504 CALL SYMBOL(HORIZ,VERT,0.2,'X(0) =',0.0,6)
505 HORIZ=STARTX+3.525
506 VERT=STARTY+1.9
507 CALL SYMBOL(HORIZ,VERT,0.2,75.0,0,-1)
508 HORIZ=STARTX+4.9
509 VERT=STARTY+1.6
510 CALL NUMBER(HORIZ,VERT,0.2,VO,0.0,4)
511 HORIZ=STARTX+0.3
512 VERT=STARTY+0.7
513 CALL PLOT(HORIZ,VERT,3)
514 HORIZ=STARTX+0.8
515 CALL PLOT(HORIZ,VERT,2)
516 HORIZ=STARTX+1.0
517 VERT=STARTY+0.6
518 CALL SYMBOL(HORIZ,VERT,0.2,'RUNGE-KUTTA',0.0,11)
519 HORIZ=STARTX+0.6
520 VERT=STARTY+0.3
521 CALL SYMBOL(HORIZ,VERT,0.2,1,0.0,-1)
522 HORIZ=STARTX+1.0
523 VERT=STARTY+0.2
524 CALL SYMBOL(HORIZ,VERT,0.2,'PIECEWISE LINEARIZATION',0.0,23)
525 NF=0
526 CALL CGPL2(XA,YA,2,NF,4,HA,HB,HC,VA,VB,VC,ALPH)
527 STOP
528 END
529 C
530 C
531 C
532 C
533 C
534 C
535 C
536 C
537 C
538 C
539 C*****
540 C
541 C   THE FOLLOWING SUBROUTINE CALCULATES THE POINT WHERE THE INTERVAL
542 C   CHORD LINE WOULD CROSS THE X-AXIS.
543 C
544 C*****
545 SUBROUTINE XPRIME(X0,X1,XSTAR)
546 IMPLICIT REAL*8 (A-H,O-Z)
547 COMMON/COEFF/A,B
548 IF(DABS(X0).LT.1.0D-08) GO TO 500
549 FO=A*X0+B*(X0**3)
550 F1=A*X1+B*(X1**3)

```



```

551      XSTAR=(X1-(F1/F0)*X0)/(1-(F1/F0))
552      GO TO 610
553 600 CONTINUE
554      XSTAR=0.00
555 610 CONTINUE
556      RETURN
557      END
558
559 C
560 C
561 C *****
562
563 C THE FOLLOWING SUBROUTINE CALCULATES THE ANGULAR FREQUENCY FOR
564 C AN INTERVAL.
565 C *****
566
567 SUBROUTINE FREQ(X0,X1,N,PSTAR)
568 IMPLICIT REAL*8 (A-H,O-Z)
569 REAL*8 N
570 COMMON/COEFF/A,B
571 F0=A*X0+B*(X0**3)
572 F1=A*X1+B*(X1**3)
573 SLOPE=(F0-F1)/(X0-X1)
574 PSTAR=DSORT(SLOPE-N**2)
575 RETURN
576 END
577
578 C
579 C
580 C *****
581
582 C THE FOLLOWING SUBROUTINE CALCULATES THE TIME INTERVAL FOR A
583 C PARTICULAR SEGMENT BY USING A NEWTON-RAPHSON METHOD. IT ALSO HAS
584 C A PROVISION FOR FINDING THE LAST TIME INTERVAL PRIOR TO THE FIRST
585 C TROUGH ON THE DISPLACEMENT-TIME CURVE.
586 C *****
587
588 SUBROUTINE TIME(X0,XDOT0,X1,DELT,XSTAR,N,P,DELTAT)
589 IMPLICIT REAL*8 (A-H,O-Z)
590 REAL*8 N
591 DELTA=DELT+0.05
592 C1=X0-XSTAR
593 C2=(N*(X0-XSTAR)+XDOT0)/P
594 C3=-N*C1+P*C2
595 C4=-P*C1-N*C2
596 C5=-(P**2-N**2)*C1-2.0*P*N*C2
597 C6=2.0*P*N*C1-(P**2-N**2)*C2
598 C *****
599
600 F=MAIN FUNCTION (DISPLACEMENT)
601 FP=F-PRIME (VELOCITY)
602
603 C *****
604 700 CONTINUE
605 F=X1-XSTAR-DEXP(-N*DELTA)*(C1*DCOS(P*DELTA)+C2*DSIN(P*DELTA))
606 FP=-DEXP(-N*DELTA)*(C3*DCOS(P*DELTA)+C4*DSIN(P*DELTA))
607 IF(FP.LT.-1.00-15) GO TO 710
608 DELTA1=DELTA-F/FP
609 DIFF=DELTA1-DELTA
610 IF(DABS(DIFF).LT.1.00-08) GO TO 730
611 DELTA=DELTA1
612 GO TO 700
613 C *****
614
615 C AS THE DISPLACEMENT APPROACHES A PEAK OR TROUGH, THE VELOCITY
616 C APPROACHES ZERO. THE RESULT IS THAT INSTEAD OF F AND FP, THE
617 C FOLLOWING ARE USED IN THE RESPECTIVE POSITIONS:
618 C
619 C FP=MAIN FUNCTION (VELOCITY)
620 C FPP=FP-PRIME (ACCELERATION)
621 C
622 C AND THE CALCULATION CONTINUES.
623 C *****
624
625 710 CONTINUE
626 DELTA=DELT+0.05
627 720 CONTINUE
628 FP=-DEXP(-N*DELTA)*(C3*DCOS(P*DELTA)+C4*DSIN(P*DELTA))
629 FPP=-DEXP(-N*DELTA)*(C5*DCOS(P*DELTA)+C6*DSIN(P*DELTA))
630 DELTA1=DELTA-FP/FPP
631 DIFF=DELTA1-DELTA
632 IF(DABS(DIFF).LT.1.00-08) GO TO 730
633 DELTA=DELTA1
634 GO TO 720
635 730 CONTINUE
636 DELTAT=DELTA
637 RETURN
638 END
639
640 C
641 C
642 C *****
643
644 C THE FOLLOWING SUBROUTINE CALCULATES THE VELOCITY AT THE END OF THE
645 C SEGMENT BEING EXAMINED.
646 C *****
647
648 SUBROUTINE SPEED(X0,XDOT0,XSTAR,N,DELTAT,P,XDOT1)
649 IMPLICIT REAL*8 (A-H,O-Z)
650 REAL*8 N
651 C1=XDOT0
652 C2=-N*(X0-XSTAR)+XDOT0)/P-P*(X0-XSTAR)
653 XDOT1=DEXP(-N*DELTAT)*(C1*DCOS(P*DELTAT)+C2*DSIN(P*DELTAT))
654 RETURN
655 END
656
657 C
658 C
659 C *****
660

```



```

861 C      THE FOLLOWING SUBROUTINE CONTAINS THE DIFFERENTIAL EQUATIONS TO BE
862 C      SOLVED BY DVERK:  THE INSTANTANEOUS VELOCITY AND THE ORIGINAL
863 C      EQUATION.  "DFORCE" IS THE DAMPING FORCE.
864 C
865 C*****
866 SUBROUTINE FCN(NEON,X,Y,YPRIME)
867 IMPLICIT REAL*8 (A-H,O-Z)
868 REAL*8 Y(NEON),YPRIME(NEON),X
869 COMMON/COEFF/A,B
870 COMMON/DAMP/C
871 YPRIME(1)=Y(2)
872 DFORCE=C*Y(2)
873 YPRIME(2)=- (DFORCE+A*Y(1)+B*(Y(1)**3))
874 RETURN
875 END

```

END OF FILE


```

1  C=====
2  C*
3  C*   PROGRAM HSPRINGO
4  C*
5  C*
6  C*
7  C*   THE FOLLOWING PROGRAM CALCULATES THE SOLUTION FOR THE
8  C*   EQUATION:
9  C*
10 C*    $X(\text{DOUBLE-DOT}) + Q = (\text{ABS}(X(\text{DOT})) ** 2) * \text{SGN}(X(\text{DOT})) + A * X + B = (X ** 3) = 0$ 
11 C*
12 C*   OBTAINED BY PIECEWISE LINEARIZATION USING CHORDS, AS WELL AS
13 C*   FINDING THE FIRST TROUGH ON THE DISPLACEMENT-TIME CURVE.
14 C*
15 C*
16 C*   THE RESULTS ARE FOUND FROM THE FOLLOWING:
17 C*
18 C*    $X = X_{\text{STAR}} + \exp(-N * (T - T_0)) * (C1 * \cos(PSTAR * (T - T_0)) + C2 * \sin(PSTAR * (T - T_0)))$ 
19 C*
20 C*
21 C*    $X(\text{DOT}) = \exp(-N * (T - T_0)) * (C3 * \cos(PSTAR * (T - T_0)) + C4 * \sin(PSTAR * (T - T_0)))$ 
22 C*
23 C*
24 C*   C1 = X0 - XSTAR
25 C*   C2 = (N * (X0 - XSTAR) + XDDOT) / P
26 C*   C3 = -N * C1 + P * C2
27 C*   C4 = -P * C1 - N * C2
28 C*
29 C*   THE VARIOUS PARAMETERS ARE EXPLAINED AND CALCULATED IN THE
30 C*   PROGRAM.
31 C*
32 C*
33 C*   THE PROGRAM ALSO SOLVES THE EQUATION USING FIFTH- AND SIXTH-
34 C*   ORDER RUNGE-KUTTA METHODS. BOTH SOLUTIONS ARE PLOTTED.
35 C*
36 C=====
37 C
38 C
39 C
40 C
41 C
42 C
43 C
44 C
45 C
46 C
47 C   IMPLICIT REAL*8 (A-H,O-Z)
48 C   EXTERNAL FCN
49 C   REAL*8 Y(2), CN(24), W(2,20)
50 C   REAL*8 N
51 C   REAL*4 AMPLP(200), AMPLRK(200), TP(200), TRK(200)
52 C   REAL*4 YP(2000), TPP(200)
53 C   REAL*4 XA(2), YA(2)
54 C   REAL*4 HA, HB, HC, VA, VB, VC
55 C   INTEGER*4 ALPH(20)
56 C   COMMON/CDEFF/A, B
57 C   COMMON/DAMP/Q
58 C=====
59 C
60 C   THE FOLLOWING VALUES ARE READ FROM DATA FILE PFITQDATA:
61 C
62 C
63 C   A, B, Q = AS IN THE EQUATION
64 C
65 C   DISPLO = INITIAL AMPLITUDE
66 C   VO = INITIAL VELOCITY
67 C   TIMEO = INITIAL TIME
68 C
69 C   H = NUMBER OF DIVISIONS OF INITIAL DISPLACEMENT
70 C   DTIME = TIME INCREMENT (RUNGE-KUTTA SOLUTION)
71 C
72 C   HA, HC = PLOT PARAMETERS FOR HORIZONTAL AXIS (HC AN INTEGRAL MULTIPLE
73 C   OF 7.0)
74 C
75 C   VC = PLOT PARAMETER FOR VERTICAL AXIS (AN INTEGRAL MULTIPLE OF 12.0,
76 C   12.0 GIVING BEST RESULTS)
77 C
78 C   XA, YA = PARAMETERS FOR PLOTTING ZERO LINE
79 C
80 C   ALPH = ARRAY FOR PLOT AXIS LABELS
81 C=====
82 C
83 C   READ(5,10) A,B,Q
84 C   10 FORMAT(3D10.6)
85 C   READ(5,20) DISPLO,VO,TIMEO
86 C   20 FORMAT(3D10.6)
87 C   READ(5,30) H,DTIME
88 C   30 FORMAT(2D10.8)
89 C   READ(5,40) HA,HC
90 C   40 FORMAT(2F10.4)
91 C   READ(5,50) VC
92 C   50 FORMAT(F10.4)
93 C   READ(5,60) XA(1)
94 C   60 FORMAT(F10.4)
95 C   READ(5,70) YA(1),YA(2)
96 C   70 FORMAT(2F10.4)
97 C   READ(5,80) (ALPH(I),I=1,12)
98 C   80 FORMAT(12A4)
99 C   READ(5,90) (ALPH(I),I=13,16)
100 C   90 FORMAT(4A4)
101 C   READ(5,100) (ALPH(I),I=17,20)
102 C   100 FORMAT(4A4)
103 C=====
104 C
105 C   C=DAMPING COEFFICIENT (INITIALLY SET AT ZERO)
106 C   N=DAMPING FACTOR
107 C
108 C=====
109 C   C=0.0
110 C   N=C/2.0

```



```

111 C *****
112 E
113 C     SETTING THE INITIAL CONDITIONS.
114 E
115 C     XO=INITIAL DISPLACEMENT FOR EACH SUCCESSIVE SEGMENT
116 C     XDOTO=INITIAL VELOCITY FOR EACH SEGMENT
117 C     XDOTP=PREVIOUS FINAL SEGMENT VELOCITY
118 C     TO=CUMULATIVE TIME TAKEN TO REACH BEGINNING OF SEGMENT
119 C     DELT=INITIAL GUESS AT TIME INTERVAL FOR THE SEGMENT BEING EXAMINED
120 C     (SEE SUBROUTINE TIME BELOW)
121 E
122 C     DELTAX=LENGTH OF SEGMENT
123 E
124 C     I=COUNTER FOR DATA POINTS TO BE PLOTTED
125 E
126 C *****
127 C     XO=DISPLO
128 C     XDOTO=VO
129 C     XDOTP=VO
130 C     TO=TIMEO
131 C     DELT=TIMEO
132 C     DELTAX=DISPLO/H
133 C     DELX=DELTAX
134 C     I=O
135 C *****
136 E
137 C     THE EQUATION PARAMETERS ARE WRITTEN INTO OUTPUT FILE PFITNUM.
138 E
139 C *****
140 C     WRITE(10,110)
141 C     110 FORMAT('1','PIECEWISE LINEAR PERIOD FOR THE EQUATION:')
142 C     WRITE(10,120)
143 C     120 FORMAT('0','X(DOUBLE-DOT)+O=(ABS(X(DOT))=2)*SGN(X(DOT))+A=X+B(X=
144 C     13)=O')
145 C     WRITE(10,130) A,B,O
146 C     130 FORMAT('0','A=',G20.10,2X,'B=',G20.10,2X,'O=',G20.10)
147 C     WRITE(10,140) DISPLO,VO
148 C     140 FORMAT('0','X(O)=' ,G20.10,2X,'X(DOT)(O)=' ,G20.10)
149 C     WRITE(10,150)
150 C     150 FORMAT('0','SX','AMPLITUDE',10X,'CUMULATIVE TIME')
151 C *****
152 E
153 C     X1=DISPLACEMENT AT END OF SEGMENT
154 E
155 C *****
156 C     X1=XO-DELTAX
157 C     160 CONTINUE
158 C     I=I+1
159 C     170 CONTINUE
160 C *****
161 E
162 C     XPRIME=SUBROUTINE FOR CALCULATING POINT WHERE SEGMENT CHORD LINE
163 C     CROSSES X-AXIS
164 C     XSTAR=CALCULATED VALUE
165 E
166 C     FREQ=SUBROUTINE FOR CALCULATING ANGULAR FREQUENCY
167 C     PSTAR=CALCULATED VALUE
168 E
169 C     TIME=SUBROUTINE FOR CALCULATING TIME INTERVAL FOR A SEGMENT
170 C     DELTAT=CALCULATED VALUE
171 E
172 C     SPEED=SUBROUTINE FOR CALCULATING SPEED AT END OF SEGMENT
173 C     XDOT1=CALCULATED VALUE
174 E
175 C     C=NOTE=THE PROGRAM STOPS IF THE DISPLACEMENT AT THE END OF A SEGMENT
176 C     IS EQUAL TO -DISPLO (I. E., UNDAMPED CASE).
177 E
178 C *****
179 C     CALL XPRIME(XO,X1,XSTAR)
180 C     CALL FREQ(XO,X1,N,PSTAR)
181 C     CALL TIME(XO,XDOTO,X1,DELT,XSTAR,N,PSTAR,DELTAT)
182 C     CALL SPEED(XO,XDOTO,XSTAR,N,DELTAT,PSTAR,XDOT1)
183 C *****
184 E
185 C     THE FOLLOWING IS THE ITERATION TO FIND THE END VELOCITY FOR THE
186 C     PRESENT SEGMENT.
187 E
188 C     DELTAV=DIFFERENCE BETWEEN XDOT1 AND XDOTP
189 C     XDOTAV=AVERAGE VELOCITY FOR SEGMENT
190 C     SGNVAV=SGN(XDOTAV)
191 E
192 C     C=NEW DAMPING COEFFICIENT FOR NEXT ITERATION
193 E
194 C *****
195 C     DELTAV=XDOT1-XDOTP
196 C     IF(DABS(DELTAV).LT.1.0D-05) GO TO 180
197 C     XDOTP=XDOT1
198 C     XDOTAV=(XDOT1+XDOTO)/2.0
199 C     SGNVAV=XDOTAV/DABS(XDOTAV)
200 C     C=O+((DABS(XDOTAV))=2)*SGNVAV/XDOTAV
201 C     N=C/2.0
202 C     GO TO 170
203 C     180 CONTINUE
204 C *****
205 E
206 C     STARTING AT STEP 210 AND CONTINUING AT 220, THE PROGRAM TESTS FOR
207 C     ZERO VELOCITY. IF IT OCCURS AT THE END OF THE SEGMENT, THE
208 C     PROGRAM STOPS (SEE STEP 320). IF NOT, THE LOCATION OF THE FIRST
209 C     TROUGH IS SOUGHT (STEPS 270 TO 290).
210 E
211 C *****
212 C     DELT=DELTAT
213 C     T1=TO+DELTAT
214 C     IF(XO.GE.O.O) GO TO 190
215 C     GO TO 210
216 C     190 CONTINUE
217 C *****
218 E
219 C     AMPLP=ARRAY FOR XO
220 C     TP=ARRAY FOR TO

```



```

221      C
222      C*****
223      AMPLP(I)=XO
224      TP(I)=TO
225      C*****
226      C
227      C   THE INTERIM VALUES ARE WRITTEN INTO OUTPUT FILE PFITONUM.
228      C
229      C*****
230      WRITE(10,200) XO,TO
231      200 FORMAT(1X,2(G20.10))
232      C*****
233      C
234      C   RESETTING THE INITIAL CONDITIONS.
235      C
236      C*****
237      XO=X1
238      X1=XO-DELTAX
239      XDOTO=XDOT1
240      TO=T1
241      GO TO 160
242      210 CONTINUE
243      IF(DABS(XDOT1).LT.1.0D-08) GO TO 220
244      GO TO 190
245      220 CONTINUE
246      C*****
247      C
248      C   THE SEGMENT IN WHICH THE FIRST TROUGH OCCURS IS WRITTEN INTO
249      C   OUTPUT FILE PFITONUM.
250      C
251      C*****
252      AMPLP(I)=XO
253      TP(I)=TO
254      WRITE(10,230) XO,TO
255      230 FORMAT(1X,2(G20.10))
256      DIFF=-DISPLO-XO
257      IF(DABS(DIFF).LT.1.0D-08) GO TO 320
258      WRITE(10,240) X1,T1
259      240 FORMAT(1X,2(G20.10))
260      WRITE(10,250)
261      250 FORMAT('-', 'TROUGH ITERATIONS')
262      WRITE(10,260)
263      260 FORMAT('0',10X,'X0',20X,'X1',14X,'X1(CALCULATED)')
264      C*****
265      C
266      C   THE FOLLOWING (ENDING AT STEP 290) IS THE CALCULATION OF THE FIRST
267      C   TROUGH. USING THE TIME INTERVAL REQUIRED TO REACH ZERO VELOCITY,
268      C   A NEW VALUE OF X1 (XF) IS CALCULATED AND COMPARED TO THE PREVIOUS
269      C   VALUE. THIS NEW VALUE BECOMES THE NEW X1 IF THE DIFFERENCE
270      C   BETWEEN XF AND THE OLD X1 IS NOT WITHIN A SET TOLERANCE, AND THE
271      C   PROCESS STARTS AGAIN.
272      C
273      C   THESE ITERATIONS ARE WRITTEN INTO OUTPUT FILE PFITONUM.
274      C
275      C*****
276      270 CONTINUE
277      C1=XO-XSTAR
278      C2=(N*(XO-XSTAR)+XDOTO)/PSTAR
279      P=PSTAR
280      XF=XSTAR+DEXP(-N*DELTAT)*(C1+DCOS(P*DELTAT)+C2*DSIN(P*DELTAT))
281      WRITE(10,280) XO,X1,XF
282      280 FORMAT(2(F12.8,2X),2X,F12.8)
283      DIFF=X1-XF
284      IF(DABS(DIFF).LT.1.0D-08) GO TO 290
285      X1=XF
286      DIFF=XO-XF
287      IF(DABS(DIFF).LT.1.0D-08) GO TO 280
288      CALL XPRIME(XO,X1,XSTAR)
289      CALL FREQ(XO,X1,N,PSTAR)
290      CALL TIME(XO,XDOTO,X1,DELT,XSTAR,N,PSTAR,DELTAT)
291      GO TO 270
292      290 CONTINUE
293      C*****
294      C
295      C   THE FINAL VALUES ARE WRITTEN INTO OUTPUT FILE HSPRINGONUM.
296      C
297      C*****
298      TF=T1
299      XF=X1
300      I=1+1
301      NPOINT=I
302      AMPLP(I)=XF
303      TP(I)=TF
304      TAU=2.0*T1
305      WRITE(10,300) DELTAX,TAU
306      300 FORMAT('-', 'DELTAX=',G20.10,2X,'PERIOD=',G20.10)
307      WRITE(10,310) XF
308      310 FORMAT('0', 'FIRST TROUGH OCCURS AT X=',G20.10)
309      GO TO 340
310      320 CONTINUE
311      C*****
312      C
313      C   THE FINAL VALUES ARE WRITTEN INTO OUTPUT FILE HSPRINGONUM (UNDOAMPED
314      C   CASE).
315      C
316      C*****
317      NPOINT=I
318      TAU=2.0*T1
319      WRITE(10,330) DELTAX,TAU
320      330 FORMAT('-', 'DELTAX=',G20.10,2X,'PERIOD=',G20.10)
321      340 CONTINUE
322      C*****
323      C
324      C   THE FOLLOWING SECTION (ENDING AT LINE 350) IS THE RUNGE-KUTTA
325      C   SOLUTION OF THE SAME ORIGINAL EQUATION.
326      C
327      C
328      C
329      C   TF=FINAL TIME
330      C   INTVL=NUMBER OF INCREMENTS

```



```

331 C
332 C #####
333 TFINAL=TFINAL+DTIME
334 INTVL=TFINAL/DTIME+1
335 C #####
336 C
337 C THE FOLLOWING ARE PARAMETERS FOR THE NUMERICAL SUBROUTINE.
338 C CONSULT THE IMSL MANUAL ON "DVERK" FOR FURTHER DETAILS.
339 C
340 C #####
341 XRK=TIME0
342 Y(1)=DISPLO
343 Y(2)=VO
344 NEON=2
345 NW=2
346 TOL=1.0D-12
347 IND=1
348 XEND=DTIME
349 C #####
350 C
351 C ZEIT=CUMULATIVE VALUE OF TIME AT END OF EACH INCREMENT
352 C
353 C #####
354 ZEIT=0.0
355 C #####
356 C
357 C AMPLRK=ARRAY FOR AMPLITUDE VALUES
358 C TRK=ARRAY FOR TIME VALUES
359 C
360 C #####
361 DO 350 J=1,INTVL
362 AMPLRK(J)=Y(1)
363 TRK(J)=ZEIT
364 CALL DVERK(NEON,FCN,XRK,Y,XEND,TOL,IND,CN,NW,W,IER)
365 C #####
366 C
367 C INCREMENTING THE TIME VALUES.
368 C
369 C #####
370 ZEIT=XEND
371 XEND=XEND+DTIME
372 350 CONTINUE
373 C #####
374 C
375 C THE CALCULATED VALUES ARE WRITTEN INTO OUTPUT FILE HSPRINGORK
376 C
377 C #####
378 WRITE(11,360)
379 360 FORMAT('1','RUNGE-KUTTA RESULTS FOR THE EQUATION:')
380 WRITE(11,370)
381 370 FORMAT('0','X(DOUBLE-DOT)+C=X(DOT)+A=X+B*(X==3)=0')
382 WRITE(11,380) A,B,C
383 380 FORMAT('0','A=',G20.10,2X,'B=',G20.10,2X,'C=',G20.10)
384 WRITE(11,390) DISPLO,VO
385 390 FORMAT('0','X(0)=',G20.10,2X,'X(DOT)(0)=',G20.10)
386 WRITE(11,400)
387 400 FORMAT('0','SX','TIME',15X,'AMPLITUDE')
388 DO 420 NO=1,INTVL
389 WRITE(11,410) TRK(NO),AMPLRK(NO)
390 410 FORMAT(1X,2(G20.10,2X))
391 420 CONTINUE
392 C #####
393 C
394 C THE NEXT FEW LINES ARE USED IN FINALIZING ALL THE PLOT PARAMETERS
395 C WHICH INCLUDES AXIS SCALING.
396 C
397 C #####
398 M=1
399 IF(DISPLO.LE.1.0) GO TO 440
400 430 CONTINUE
401 DECD=DISPLO/(10.0==M)
402 IF(DECD.LE.1.0) GO TO 460
403 M=M+1
404 GO TO 430
405 440 CONTINUE
406 DIGD=DISPLO*(10.0==M)
407 IF(DIGD.GE.1.0) GO TO 450
408 M=M+1
409 GO TO 440
410 450 CONTINUE
411 DECD=DIGD/10.0
412 M=-(M-1)
413 460 CONTINUE
414 IF(DECD.LE.0.1) VFACT=0.1
415 IF((DECD.GT.0.1).AND.(DECD.LE.0.25)) VFACT=0.25
416 IF((DECD.GT.0.25).AND.(DECD.LE.0.5)) VFACT=0.5
417 IF((DECD.GT.0.5).AND.(DECD.LE.1.0)) VFACT=1.0
418 SFACT=VC/10.0
419 VA=-VFACT*SFACT*10.0==M
420 DEN=VC/(2.0*SFACT)
421 VB=(VFACT/DEN)*10.0==M
422 C
423 M=1
424 IF(TFINAL.LE.1.0) GO TO 480
425 470 CONTINUE
426 DECT=TFINAL/(10.0==M)
427 IF(DECT.LE.1.0) GO TO 500
428 M=M+1
429 GO TO 470
430 480 CONTINUE
431 DIGT=TFINAL*(10.0==M)
432 IF(DIGT.GE.1.0) GO TO 490
433 M=M+1
434 GO TO 480
435 490 CONTINUE
436 DECT=DIGT/10.0
437 M=-(M-1)
438 500 CONTINUE
439 IF(DECT.LE.0.07) HFACT=0.1
440 IF((DECT.GT.0.07).AND.(DECT.LE.0.14)) HFACT=0.2

```



```

441      IF((DECT.GT.0.14).AND.(DECT.LE.0.35)) HFACT=0.5
442      IF((DECT.GT.0.35).AND.(DECT.LE.0.7)) HFACT=1.0
443      IF((DECT.GT.0.7).AND.(DECT.LE.1.0)) HFACT=2.0
444      HB=HFACT*10.0*(M-1)
445      LONG=HC
446      XA(2)=DFLOAT(LONG)*HB-0.005
447      C*****
448      C
449      C      THE FOLLOWING SECTION SETS UP THE PIECEWISE LINEAR RESULTS SO THAT
450      C      NO MORE THAN 13 POINTS ARE PLOTTED FOR A CLEARER PRESENTATION.
451      C
452      C      YP=ARRAY FOR PLOTTED PIECEWISE LINEAR DISPLACEMENTS
453      C      TPP=ARRAY FOR PLOTTED PIECEWISE LINEAR TIME VALUES
454      C
455      C*****
456      COUNT=DFLOAT(NPOINT)
457      INC=COUNT/10.0
458      DELINC=COUNT/10.0-DFLOAT(INC)
459      IF(DELINC.GE.0.5) INC=INC+1
460      NP=1
461      ND=1
462      510 CONTINUE
463      YP(NP)=AMPLP(ND)
464      TPP(NP)=TP(ND)
465      NP=NP+1
466      ND=ND+INC
467      IF(ND.GE.NPOINT) GO TO 520
468      GO TO 510
469      520 CONTINUE
470      YP(NP)=AMPLP(NPOINT)
471      TPP(NP)=TP(NPOINT)
472      C*****
473      C
474      C      PLOTTING THE SOLUTIONS. CONSULT THE WRITEUP ON CGPL/CGPL2 AND
475      C      THE MANUAL ON DIGITAL PLOTTING FOR DETAILS.
476      C
477      C*****
478      ND=INTVL
479      NF=1
480      CALL CGPL2(TRK,AMPLRK,ND,NF,5,HA,HB,HC,VA,VB,VC,ALPH)
481      ND=NP
482      CALL CGPL2(TPP,YP,ND,2,1,HA,HB,HC,VA,VB,VC,ALPH)
483      NF=4
484      CALL CGPL2(XA,YA,2,NF,4,HA,HB,HC,VA,VB,VC,ALPH)
485      STARTX=(HC-6.3)/2.0
486      STARTY=VC+0.5
487      HORIZ=HA
488      VERT=VC+0.5
489      CALL PLOT(HORIZ,VERT,3)
490      HORIZ=HC
491      CALL PLOT(HORIZ,VERT,2)
492      VERT=VERT+4.6
493      CALL PLOT(HORIZ,VERT,2)
494      HORIZ=HA
495      CALL PLOT(HORIZ,VERT,2)
496      VERT=VC+0.5
497      CALL PLOT(HORIZ,VERT,2)
498      HORIZ=STARTX+0.2
499      VERT=STARTY+4.2
500      CALL SYMBOL(HORIZ,VERT,0.2,'A ',0.0,3)
501      HORIZ=STARTX+1.0
502      CALL NUMBER(HORIZ,VERT,0.2,A,0.0,4)
503      HORIZ=STARTX+3.0
504      CALL SYMBOL(HORIZ,VERT,0.2,'B ',0.0,3)
505      HORIZ=STARTX+3.8
506      CALL NUMBER(HORIZ,VERT,0.2,B,0.0,4)
507      HORIZ=STARTX+2.0
508      VERT=STARTY+3.8
509      CALL SYMBOL(HORIZ,VERT,0.2,'Q ',0.0,3)
510      HORIZ=STARTX+2.8
511      CALL NUMBER(HORIZ,VERT,0.2,Q,0.0,4)
512      HORIZ=STARTX+0.2
513      VERT=STARTY+3.1
514      CALL SYMBOL(HORIZ,VERT,0.2,52,0.0,-1)
515      HORIZ=STARTX+0.4
516      CALL SYMBOL(HORIZ,VERT,0.2,'X (PIECEWISE',0.0,11)
517      HORIZ=STARTX+1.0
518      VERT=STARTY+2.8
519      CALL SYMBOL(HORIZ,VERT,0.2,'LINEARIZATION) ',0.0,15)
520      HORIZ=STARTX+4.4
521      CALL NUMBER(HORIZ,VERT,0.2,DELX,0.0,4)
522      HORIZ=STARTX+0.2
523      VERT=STARTY+2.4
524      CALL SYMBOL(HORIZ,VERT,0.2,52,0.0,-1)
525      HORIZ=STARTX+0.4
526      CALL SYMBOL(HORIZ,VERT,0.2,'T (RUNGE-KUTTA) ',0.0,17)
527      HORIZ=STARTX+4.0
528      CALL NUMBER(HORIZ,VERT,0.2,DTIME,0.0,4)
529      HORIZ=STARTX+0.2
530      VERT=STARTY+1.6
531      CALL SYMBOL(HORIZ,VERT,0.2,'X(O) ',0.0,6)
532      HORIZ=STARTX+1.6
533      CALL NUMBER(HORIZ,VERT,0.2,DISPLO,0.0,4)
534      HORIZ=STARTX+3.5
535      CALL SYMBOL(HORIZ,VERT,0.2,'X(O) ',0.0,6)
536      HORIZ=STARTX+3.525
537      VERT=STARTY+1.9
538      CALL SYMBOL(HORIZ,VERT,0.2,75,0.0,-1)
539      HORIZ=STARTX+4.9
540      VERT=STARTY+1.6
541      CALL NUMBER(HORIZ,VERT,0.2,VO,0.0,4)
542      HORIZ=STARTX+0.3
543      VERT=STARTY+0.7
544      CALL PLOT(HORIZ,VERT,3)
545      HORIZ=STARTX+0.8
546      CALL PLOT(HORIZ,VERT,2)
547      HORIZ=STARTX+1.0
548      VERT=STARTY+0.6
549      CALL SYMBOL(HORIZ,VERT,0.2,'RUNGE-KUTTA',0.0,11)
550      HORIZ=STARTX+0.6

```



```

551      VERT=STARTY+0.3
552      CALL SYMBOL(HORIZ,VERT,0.2,1,0.0,-1)
553      HORIZ=STARTX+1.0
554      VERT=STARTY+0.2
555      CALL SYMBOL(HORIZ,VERT,0.2,'PIECEWISE LINEARIZATION',0.0,23)
556      NF=0
557      CALL CGPL2(XA,YA,2,NF,4,HA,HB,HC,VA,YB,VC,ALPH)
558      STOP
559      END
560
561      C
562      C
563      C
564      C
565      C
566      C
567      C
568      C
569      C
570      C
571      C
572      C      THE FOLLOWING SUBROUTINE CALCULATES THE POINT WHERE THE INTERVAL
573      C      CHORD LINE WOULD CROSS THE X-AXIS.
574      C
575      C
576      C      SUBROUTINE XPRIME(X0,X1,XSTAR)
577      C      IMPLICIT REAL=8 (A-H,O-Z)
578      C      COMMON/COEFF/A,B
579      C      IF(DABS(X0).LT.1.0D-08) GO TO 600
580      C      FO=A+X0+B*(X0**3)
581      C      F1=A+X1+B*(X1**3)
582      C      XSTAR=(X1-(F1/FO)*X0)/(1-(F1/FO))
583      C      GO TO 610
584      C      600 CONTINUE
585      C      XSTAR=0.0D0
586      C      610 CONTINUE
587      C      RETURN
588      C      END
589      C
590      C
591      C
592      C
593      C
594      C      THE FOLLOWING SUBROUTINE CALCULATES THE ANGULAR FREQUENCY FOR
595      C      AN INTERVAL.
596      C
597      C
598      C      SUBROUTINE FREQ(X0,X1,N,PSTAR)
599      C      IMPLICIT REAL=8 (A-H,O-Z)
600      C      REAL=8 N
601      C      COMMON/COEFF/A,B
602      C      FO=A+X0+B*(X0**3)
603      C      F1=A+X1+B*(X1**3)
604      C      SLOPE=(FO-F1)/(X0-X1)
605      C      PSTAR=DSORT(SLOPE-N**2)
606      C      RETURN
607      C      END
608      C
609      C
610      C
611      C
612      C
613      C      THE FOLLOWING SUBROUTINE CALCULATES THE TIME INTERVAL FOR A
614      C      PARTICULAR SEGMENT BY USING A NEWTON-RAPHSON METHOD. IT ALSO HAS
615      C      A PROVISION FOR FINDING THE LAST TIME INTERVAL PRIOR TO THE FIRST
616      C      TROUGH ON THE DISPLACEMENT-TIME CURVE.
617      C
618      C
619      C      SUBROUTINE TIME(X0,XDOT0,X1,DELT,XSTAR,N,P,DELTAT)
620      C      IMPLICIT REAL=8 (A-H,O-Z)
621      C      REAL=8 N
622      C      DELTA=DELT+0.05
623      C      C1=X0-XSTAR
624      C      C2=(N*(X0-XSTAR)+XDOT0)/P
625      C      C3=-N*C1+P*C2
626      C      C4=-P*C1-N*C2
627      C      C5=-(P**2-N**2)*C1-2.0*P*N*C2
628      C      C6=2.0*P*N*C1-(P**2-N**2)*C2
629      C
630      C
631      C      F=MAIN FUNCTION (DISPLACEMENT)
632      C      FP=F-PRIME (VELOCITY)
633      C
634      C
635      C      700 CONTINUE
636      C      F=X1-XSTAR-DEXP(-N*DELTA)*(C1*DCOS(P*DELTA)+C2*DSIN(P*DELTA))
637      C      FP=-DEXP(-N*DELTA)*(C3*DCOS(P*DELTA)+C4*DSIN(P*DELTA))
638      C      IF(FP.LT.-1.0D-15) GO TO 710
639      C      DELTA1=DELTA-F/FP
640      C      DIFF=DELTA1-DELTA
641      C      IF(DABS(DIFF).LT.1.0D-08) GO TO 730
642      C      DELTA=DELTA1
643      C      GO TO 700
644      C
645      C
646      C      AS THE DISPLACEMENT APPROACHES A PEAK OR TROUGH, THE VELOCITY
647      C      APPROACHES ZERO. THE RESULT IS THAT INSTEAD OF F AND FP, THE
648      C      FOLLOWING ARE USED IN THE RESPECTIVE POSITIONS:
649      C
650      C      FPP=MAIN FUNCTION (VELOCITY)
651      C      FPP=F-PRIME (ACCELERATION)
652      C
653      C      AND THE CALCULATION CONTINUES.
654      C
655      C
656      C      710 CONTINUE
657      C      DELTA=DELT+0.05
658      C      720 CONTINUE
659      C      FP=-DEXP(-N*DELTA)*(C3*DCOS(P*DELTA)+C4*DSIN(P*DELTA))
660      C      FPP=-DEXP(-N*DELTA)*(C5*DCOS(P*DELTA)+C6*DSIN(P*DELTA))

```



```

861      DELTA1=DELTA-FP/FPP
862      DIFF=DELTA1-DELTA
863      IF(DABS(DIFF).LT.1.0D-08) GO TO 730
864      DELTA=DELTA1
865      GO TO 720
866 730  DELTAT=DELTA
867      RETURN
868      END
869
870  C
871  C
872  C*****
873  C
874      THE FOLLOWING SUBROUTINE CALCULATES THE VELOCITY AT THE END OF THE
875      SEGMENT BEING EXAMINED.
876  C
877  C*****
878      SUBROUTINE SPEED(XO,XDOTO,XSTAR,N,DELTAT,P,XDOT1)
879      IMPLICIT REAL*8 (A-H,O-Z)
880      REAL*8 N
881      C1=XDOTO
882      C2=-N*(N*(XO-XSTAR)+XDOTO)/P-P*(XO-XSTAR)
883      XDOT1=DEXP(-N*DELTAT)*(C1=DCOS(P*DELTAT)+C2=DSIN(P*DELTAT))
884      RETURN
885      END
886
887  C
888  C
889  C*****
890  C
891      THE FOLLOWING SUBROUTINE CONTAINS THE DIFFERENTIAL EQUATIONS TO BE
892      SOLVED BY DVERK: THE INSTANTANEOUS VELOCITY AND THE ORIGINAL
893      EQUATION. "DFORCE" IS THE DAMPING FORCE.
894  C
895  C*****
896      SUBROUTINE FCN(N,X,Y,YPRIME)
897      IMPLICIT REAL*8 (A-H,O-Z)
898      REAL*8 Y(N),YPRIME(N),X
899      COMMON/COEFF/A,B
900      COMMON/DAMP/O
901      YPRIME(1)=Y(2)
902      IF(DABS(YPRIME(1)).LT.1.0D-08) GO TO 800
903      SGNV=YPRIME(1)/DABS(YPRIME(1))
904      GO TO 810
905 800  CONTINUE
906      SGNV=0.0D0
907 810  CONTINUE
908      DFORCE=0*(DABS(YPRIME(1))==2)=SGNV
909      YPRIME(2)=- (DFORCE+A*Y(1)+B*(Y(1)**3))
910      RETURN
911      END

```

END OF FILE


```

1  C*****
2  C*
3  C*   PROGRAM VANDERPOL
4  C*
5  C*
6  C*
7  C*   THE FOLLOWING PROGRAM CALCULATES THE SOLUTION FOR THE VAN DER
8  C*   POL EQUATION:
9  C*
10 C*    $X(\text{DOUBLE-DOT}) + \mu X' = (X^2 - 1)X(\text{DOT}) + X = 0$ 
11 C*
12 C*   OBTAINED BY PIECEWISE LINEARIZATION USING CHORDS OVER SEVERAL
13 C*   CYCLES OF THE DISPLACEMENT-TIME CURVE.
14 C*
15 C*
16 C*   THE RESULTS ARE BASED ON THE STANDARD FORM:
17 C*
18 C*    $X = C \exp(R \cdot \text{DELTA}T)$ 
19 C*
20 C*    $R1 = -N + \text{SORT}(N^2 - P^2)$ 
21 C*    $R2 = -N - \text{SORT}(N^2 - P^2)$ 
22 C*
23 C*   WHERE THE UNDERDAMPED, CRITICALLY DAMPED, AND OVERDAMPED
24 C*   APPROXIMATIONS WOULD DEPEND UPON MU AND AN ESTIMATED
25 C*   DISPLACEMENT.
26 C*
27 C*   THE VARIOUS PARAMETERS ARE EXPLAINED AND CALCULATED IN THE
28 C*   PROGRAM.
29 C*
30 C*
31 C*   THE PROGRAM ALSO SOLVES THE EQUATION USING FIFTH- AND SIXTH-
32 C*   ORDER RUNGE-KUTTA METHODS. BOTH SOLUTIONS ARE PLOTTED, AS
33 C*   WELL AS THE PHASE-PLANE DIAGRAMS.
34 C*
35 C*****
36
37
38
39
40
41
42
43
44
45
46 IMPLICIT REAL*8 (A-H,O-Z)
47 EXTERNAL FCN
48 REAL*8 Y(2), CN(24), W(2,20)
49 REAL*8 K, MU, N
50 REAL*4 AMPLP(10000), VP(10000), TP(10000)
51 REAL*4 AMPLRK(10000), VRK(10000), TRK(10000)
52 REAL*4 YP(10000), YPP(10000), TPP(10000)
53 REAL*4 YRK(10000), VPRK(10000), TPRK(10000)
54 REAL*4 HAD, HBD, HCD, YAD, VSD, YCD
55 REAL*4 HAP, HBP, HCP, YAP, VBP, YCP
56 REAL*4 XAD(2), YAD(2), XAPH(2), YAPH(2), XAPV(2), YAPV(2)
57 INTEGER*4 ALPHD(20), ALPHP(20)
58 COMMON MU
59 C*****
60
61 THE FOLLOWING VALUES ARE READ FROM DATA FILE VOPDATA:
62
63
64 MU=AS IN THE EQUATION
65
66 DISPLO=INITIAL AMPLITUDE
67 VO=INITIAL VELOCITY
68 TIMEO=INITIAL TIME
69
70 DELTAT=TIME INCREMENT (PIECEWISE SOLUTION)
71 HCYCLE=NUMBER OF HALF-CYCLES (PIECEWISE SOLUTION)
72 DT=TIME INCREMENT (RUNGE-KUTTA SOLUTION)
73
74 HAD,HCD=PLOT PARAMETERS FOR HORIZONTAL AXIS OF DISPLACEMENT-TIME
75 CURVE (HCD AN INTEGRAL MULTIPLE OF 7.0)
76
77 VCD=PLOT PARAMETER FOR VERTICAL AXIS OF DISPLACEMENT-TIME CURVE
78 (AN INTEGRAL MULTIPLE OF 12.0, 12.0 GIVING BEST RESULTS)
79
80 XAD,YAD=PLOT PARAMETERS FOR PLOTTING ZERO LINE OF DISPLACEMENT-
81 TIME CURVE
82
83 ALPHD=ARRAY FOR PLOT AXIS LABELS OF DISPLACEMENT-TIME CURVE
84
85 VEP=PLOT PARAETER FOR VERTICAL AXIS OF PHASE-PLAE DIAGRAM (AN
86 INTEGRAL MULTIPLE OF 12.0, 12.0 GIVING BEST RESULTS)
87
88 ALPHP=ARRAY FOR PLOT AXIS LABELS OF PHASE-PLANE DIAGRAM
89 C*****
90
91 READ(5,10) MU
92 10 FORMAT(D10.6)
93 READ(5,20) DISPLO,VO,TIMEO
94 20 FORMAT(3D10.6)
95 READ(5,30) DELTAT,HCYCLE,DT
96 30 FORMAT(3D10.6)
97 READ(5,40) HAD,HCD
98 40 FORMAT(3F10.4)
99 READ(5,50) VCD
100 50 FORMAT(F10.4)
101 READ(5,60) XAD(1)
102 60 FORMAT(F10.4)
103 READ(5,70) YAD(1),YAD(2)
104 70 FORMAT(2F10.4)
105 READ(5,80) (ALPHD(I),I=1,12)
106 80 FORMAT(12A4)
107 READ(5,90) (ALPHD(I),I=13,16)
108 90 FORMAT(4A4)
109 READ(5,100) (ALPHD(I),I=17,20)
110 100 FORMAT(4A4)

```



```

111      READ(5,110) VCP
112      110 FORMAT(F10.4)
113      READ(5,120) (ALPHP(I),I=1,12)
114      120 FORMAT(12A4)
115      READ(5,130) (ALPHP(I),I=13,16)
116      130 FORMAT(4A4)
117      READ(5,140) (ALPHP(I),I=17,20)
118      140 FORMAT(4A4)
119      C *****
120      E
121      C      SETTING THE INITIAL CONDITIONS
122      C
123      C      XO=INITIAL DISPLACEMENT FOR EACH SUCCESSIVE SEGMENT
124      C      XDOTO=INITIAL VELOCITY FOR EACH SEGMENT
125      C      TO=INITIAL TIME FOR EACH SEGMENT
126      C      XPREV=PREVIOUS FINAL SEGMENT DISPLACEMENT
127      C
128      C *****
129      C      XO=DISPLO
130      C      XDOTO=VO
131      C      TO=TIMEO
132      C      XPREV=DISPLO
133      C *****
134      E
135      C      SIG=SUBROUTINE FOR CALCULATING SGN(X)
136      C      SGNXO=CALCULATED VALUE
137      C
138      C *****
139      C      CALL SIG(XO,SGNXO)
140      C *****
141      C
142      C      I=COUNTER USED FOR PLOTTING PURPOSES
143      C
144      C      K=COUNTER FOR THE NUMBER OF HALF-CYCLES (AN INTEGRAL MULTIPLE OF
145      C      10.0, 10.0 GIVING BEST RESULTS)
146      C
147      C *****
148      C      I=0
149      C      K=1.0
150      C *****
151      E
152      C      THE EQUATION PARAMETERS ARE WRITTEN INTO OUTPUT FILE VDPNUM.
153      C
154      C *****
155      C      WRITE(10,150)
156      C      150 FORMAT('1','PIECEWISE LINEAR RESULTS FOR THE EQUATION:')
157      C      WRITE(10,160)
158      C      160 FORMAT('0','X(DOUBLE-DOT)+MU*(X==2-1)=X(DOT)+X=0')
159      C      WRITE(10,170) MU
160      C      170 FORMAT('0','MU=',G20.10)
161      C      WRITE(10,180) DISPLO,VO
162      C      180 FORMAT('0','X(0)=',G20.10,2X,'X(DOT)(0)=',G20.10)
163      C      WRITE(10,190)
164      C      190 FORMAT('0',9X,'TIME',19X,'X',18X,'VELOCITY')
165      C *****
166      C
167      C      THE FOLLOWING IS THE CALCULATION OF THE SOLUTION FOR A PARTICULAR
168      C      HALF-CYCLE, STARTING AT STEP 200 AND ENDING AT STEP 250
169      C
170      C      X=DISPLACEMENT USED FOR CALCULATING A DAMPING COEFFICIENT (SEE
171      C      BELOW)
172      C      T1=CUMULATIVE TIME
173      C
174      C *****
175      C      200 CONTINUE
176      C      I=I+1
177      C      X=XO
178      C      T1=TO+DELTAT
179      C *****
180      E
181      C      THE FOLLOWING IS THE ITERATION TO FIND THE END DISPLACEMENT FOR
182      C      THE PRESENT SEGMENT.
183      C
184      C      C=DAMPING COEFFICIENT
185      C      N=DAMPING FACTOR
186      C
187      C      DAMP=SUBROUTINE FOR CALCULATING DEGREE OF DAMPING AND
188      C      CORRESPONDING DAMPED ANGULAR FREQUENCY
189      C      PSTAR=CALCULATED VALUE
190      C
191      C      DISPL=SUBROUTINE FOR CALCULATING SEGMENT LENGTH
192      C      DELTAX=CALCULATED VALUE
193      C
194      C      SPEED=SUBROUTINE FOR CALCULATING END VELOCITY
195      C      XDOT1=CALCULATED VALUE
196      C
197      C      X1=SEGMENT END
198      C
199      C *****
200      C      210 CONTINUE
201      C      C=MU*(X==2-1.0)
202      C      N=C/2.0
203      C      CALL DAMP(N,PSTAR)
204      C      CALL DISPL(XO,XDOTO,N,DELTAT,PSTAR,DELTAX)
205      C      CALL SPEED(XO,XDOTO,N,DELTAT,PSTAR,XDOT1)
206      C      X1=XO+DELTAX
207      C      DIFF=X1-XPREV
208      C      IF(DABS(DIFF).LT.1.0D-08) GO TO 220
209      C      XPREV=X1
210      C      X=XO+DELTAX/2.0
211      C      GO TO 210
212      C      220 CONTINUE
213      C *****
214      E
215      C      TESTING FOR THE CHANGE IN SIGN OF THE DISPLACEMENT. IF IT IS
216      C      DIFFERENT, THE PROGRAM BEGINS TESTING FOR EITHER A PEAK OR TROUGH,
217      C      STARTING AT STEP 250. IF NOT, THE ITERATION CONTINUES
218      C
219      C *****
220      C      CALL SIG(X1,SGNX1)

```



```

221      IF(SGNX1.NE.SGNX0) GO TO 250
222      230 CONTINUE
223      C *****
224      C
225      C      TP=ARRAY FOR TO
226      C      AMPLP=ARRAY FOR XO
227      C      VP=ARRAY FOR XDOTO
228      C
229      C *****
230      C      TP(I)=TO
231      C      AMPLP(I)=XO
232      C      VP(I)=XDOTO
233      C *****
234      C
235      C      THE INTERIM VALUES ARE WRITTEN INTO OUTPUT FILE VDPNUM.
236      C
237      C *****
238      C      WRITE(10,240) TO,XO,XDOTO
239      C      240 FORMAT(1X,3(G20.10,2X))
240      C *****
241      C
242      C      RESETTING THE INITIAL CONDITIONS FOR THE NEXT SEGMENT.
243      C
244      C *****
245      C      XO=X1
246      C      XDOTO=XDOT1
247      C      TO=T1
248      C      GO TO 200
249      C *****
250      C
251      C      STARTING AT STEP 250, THE PROGRAM TESTS FOR EITHER A PEAK OR
252      C      TROUGH (I. E., ZERO VELOCITY). IF THE VELOCITY IS ZERO AT THE END
253      C      OF THE CURRENT TIME INTERVAL, THE PROGRAM MOVES TO STEP 370 AND
254      C      STARTS THE CALCULATIONS FOR THE NEXT HALF-CYCLE. IF NOT, A
255      C      POSSIBLE CHANGE OF SIGN FOR THE VELOCITY IS TESTED FOR. SHOULD IT
256      C      BE NEGATIVE, THE PROGRAM GOES TO STEP 230 AND CONTINUES. A
257      C      POSITIVE RESULT ALLOWS THE PROGRAM TO ESTIMATE THE VALUE OF THE
258      C      DISPLACEMENT AT THIS PEAK OR TROUGH, XF (SEE STEPS 310 TO 320).
259      C      THIS VALUE IS COMPARED TO A PREVIOUS VALUE (XPREV), AND IF THE
260      C      DIFFERENCE IS VERY SMALL, THE PROGRAM GOES TO STEP 320 AND
261      C      CONTINUES. IF NOT, XF BECOMES XPREV, THE PARAMETERS ARE RESET,
262      C      AND THE ITERATION CONTINUES AT STEP 310.
263      C *****
264      C
265      C      250 CONTINUE
266      C      IF(DABS(XDOT1).GT.1.0D-08) GO TO 290
267      C      I=I+1
268      C      TP(I)=T1
269      C      AMPLP(I)=X1
270      C      VP(I)=XDOT1
271      C      WRITE(10,260)
272      C      260 FORMAT('O')
273      C      WRITE(10,270) T1,X1,XDOT1
274      C      270 FORMAT(1X,3(G20.10,2X))
275      C      WRITE(10,280)
276      C      280 FORMAT('O')
277      C      SGNX0=-SGNX0
278      C      GO TO 370
279      C      290 CONTINUE
280      C      CALL DISPL(XO,XDOTO,N,DELTAT,PSTAR,DELX)
281      C      CALL SPEED(XO,XDOTO,N,DELTAT,PSTAR,XDOT1)
282      C      CALL SIG(XDOT1,SGNV1)
283      C      IF(SGNV1.EQ.SGNX1) GO TO 230
284      C      TP(I)=TO
285      C      AMPLP(I)=XO
286      C      VP(I)=XDOTO
287      C      WRITE(10,300) TO,XO,XDOTO
288      C      300 FORMAT(1X,3(G20.10,2X))
289      C      XF=XO-DELX
290      C      T1=T1-DELTAT
291      C *****
292      C
293      C      THE ITERATION FOR EITHER THE PEAK OR TROUGH IS DONE FROM STEPS 310
294      C      TO 320.
295      C
296      C      DELT=ESTIMATED VALUE FOR THE TIME INTERVAL TO REACH ZERO VELOCITY
297      C      TIME=SUBROUTINE FOR CALCULATING THE PRECISE VALUE FOR THIS TIME
298      C      INTERVAL
299      C      DTIME=CALCULATED VALUE
300      C
301      C *****
302      C      310 CONTINUE
303      C      DELT=DELTAT
304      C      CALL TIME(XO,XDOTO,DELT,N,PSTAR,DTIME)
305      C      CALL DISPL(XO,XDOTO,N,DTIME,PSTAR,DELTA)
306      C      CALL SPEED(XO,XDOTO,N,DTIME,PSTAR,XDOT1)
307      C      X1=XO+DELTA
308      C      DELX=X1-XF
309      C      IF(DABS(DELX).LT.1.0D-08) GO TO 320
310      C      XF=X1
311      C      N=MU*(XF**2-1.0)/2.0
312      C      CALL DAMP(N,PSTAR)
313      C      GO TO 310
314      C      320 CONTINUE
315      C *****
316      C
317      C      THE VALUES FOR THE PEAK OR TROUGH ARE WRITTEN INTO OUTPUT FILE
318      C      VDPNUM.
319      C
320      C *****
321      C      T1=T1+DTIME
322      C      CALL SPEED(XO,XDOTO,N,DTIME,PSTAR,XDOT1)
323      C      I=I+1
324      C      TP(I)=T1
325      C      AMPLP(I)=X1
326      C      VP(I)=XDOT1
327      C      WRITE(10,330)
328      C      330 FORMAT('O')
329      C      WRITE(10,340) T1,X1,XDOT1
330      C      340 FORMAT(1X,3(G20.10,2X))

```



```

331      WRITE(10,350)
332      350 FORMAT(1X,3(G20.10,2X))
333      C *****
334      C
335      C      TESTING FOR THE NUMBER OF HALF-CYCLES.
336      C
337      C *****
338      IF(K.EQ.HCYCLE) GO TO 380
339      C *****
340      C
341      C      RESETTING THE INITIAL CONDITIONS FOR THE NEXT HALF-CYCLE.
342      C
343      C *****
344      XO=X1
345      XDOTO=XDOT1
346      XPREV=XO
347      X=XO
348      SGNXO=-SGNXO
349      K=K+1.0
350      C *****
351      C
352      C      ONCE THE PEAK OR TROUGH CO-ORDINATES HAVE BEEN FOUND, THE VALUES
353      C      AT THE END OF THE NEXT TIME INCREMENT ARE CALCULATED, SINCE THE
354      C      FORMER DO NOT OCCUR AT THE END OF THE PREVIOUS ONE. THE
355      C      DISPLACEMENT IS FOUND EXACTLY AS WAS DONE FROM STEPS 210 TO 220,
356      C      EXCEPT WITH THE TIME INTERVAL BEING DIFFT.
357      C *****
358      DIFFT=DELTAT-DTIME
359      TO=T1+DIFFT
360      C
361      360 CONTINUE
362      C=MU*(X==2-1.0)
363      N=C/2.0
364      CALL DAMP(N,PSTAR)
365      CALL DISPL(XO,XDOTO,N,DIFFT,PSTAR,DELTAX)
366      CALL SPEED(XO,XDOTO,N,DIFFT,PSTAR,XDOT1)
367      X1=XO+DELTAX
368      DIFFX=X1-XPREV
369      IF(DABS(DIFFX).LT.1.0D-08) GO TO 370
370      XPREV=X1
371      X=XO+DELTAX/2.0
372      GO TO 360
373      370 CONTINUE
374      XO=X1
375      XDOTO=XDOT1
376      GO TO 200
377      380 CONTINUE
378      NPOINT=I
379      C *****
380      C
381      C      THE FOLLOWING SECTION (ENDING AT LINE 390) IS THE RUNGE-KUTTA
382      C      SOLUTION OF THE SAME ORIGINAL EQUATION.
383      C
384      C
385      C
386      C      TFINAL=FINAL TIME
387      C      INTVL=NUMBER OF INCREMENTS
388      C
389      C *****
390      TFINAL=T1+0.1
391      INTVL=TFINAL/DT+1
392      C *****
393      C
394      C      THE FOLLOWING ARE PARAMETERS FOR THE RUNGE-KUTTA SUBROUTINE.
395      C      CONSULT THE IMSL MANUAL FOR "OVERK" FOR FURTHER DETAILS.
396      C
397      C *****
398      XRK=TIMEO
399      Y(1)=DISPLO
400      Y(2)=VO
401      C *****
402      C
403      C      BIGRK=LARGEST VALUE OF RUNGE-KUTTA VELOCITY (FOR PLOT SCALING)
404      C
405      C *****
406      NEON=2
407      NW=2
408      TOL=1.0D-04
409      IND=1
410      XEND=DT
411      C *****
412      C
413      C      ZEIT=CUMULATIVE VALUE OF TIME AT END OF EACH INCREMENT
414      C
415      C *****
416      ZEIT=0.0
417      C *****
418      C
419      C      AMPLRK=ARRAY FOR AMPLITUDE VALUES
420      C      TRK=ARRAY FOR TIME VALUES
421      C      VRK=ARRAY FOR VELOCITY VALUES
422      C
423      C *****
424      DO 390 J=1,INTVL
425      AMPLRK(J)=Y(1)
426      TRK(J)=ZEIT
427      VRK(J)=Y(2)
428      CALL DVERK(NEON,FCN,XRK,Y,XEND,TOL,IND,CN,NW,W,IER)
429      C *****
430      C
431      C      INCREMENTING THE TIME VALUES.
432      C
433      C *****
434      ZEIT=XEND
435      XEND=XEND+DT
436      390 CONTINUE
437      C *****
438      C
439      C      THE CALCULATED VALUES ARE WRITTEN INTO OUTPUT FILE VDPNNUM.
440      C

```



```

441 C *****
442 WRITE(11,400)
443 400 FORMAT('1','RUNGE-KUTTA RESULTS FOR THE EQUATION:')
444 WRITE(11,410)
445 410 FORMAT('0','X(DOUBLE-DOT)+MU=(X**2-1)*X(DOT)+X**0')
446 WRITE(11,420) MU
447 420 FORMAT('0','MU=',G20.10)
448 WRITE(11,430) DISPLO,V0
449 430 FORMAT('0','X(0)=',G20.10,2X,'X(DOT)(0)=',G20.10)
450 WRITE(11,440)
451 440 FORMAT('1',4X,'TIME',10X,'X(T)',8X,'VELOCITY')
452 DO 460 M=1,INTVL
453 WRITE(11,450) TRK(M),AMPLRK(M),VRK(M)
454 450 FORMAT(1X,3(G20.10,2X))
455 460 CONTINUE
456 C *****
457 E
458 E THE FOLLOWING FINDS THE LARGEST MAGNITUDES OF THE DISPLACEMENTS
459 E AND VELOCITIES FOR THE SOLUTIONS FOR PLOT SCALING PURPOSES.
460 E
461 C BIGDP=LARGEST VALUE OF PIECEWISE LINEAR DISPLACEMENT
462 C BIGVP=LARGEST VALUE OF PIECEWISE LINEAR VELOCITY
463 C
464 C BIGDRK=LARGEST VALUE OF RUNGE-KUTTA DISPLACEMENT
465 C BIGVRK=LARGEST VALUE OF RUNGE-KUTTA VELOCITY
466 C
467 C *****
468 BIGDP=DISPLO
469 BIGVP=V0
470 DO 470 IB=1,NPOINT
471 DISP=AMPLP(IB)
472 VEL=VP(IB)
473 IF(DABS(DISP).GE.BIGDP) BIGDP=DABS(DISP)
474 IF(DABS(VEL).GE.BIGVP) BIGVP=DABS(VEL)
475 470 CONTINUE
476 BIGDRK=DISPLO
477 BIGVRK=V0
478 DO 480 IB=1,INTVL
479 DISP=AMPLRK(IB)
480 VEL=VRK(IB)
481 IF(DABS(DISP).GE.BIGDRK) BIGDRK=DABS(DISP)
482 IF(DABS(VEL).GE.BIGVRK) BIGVRK=DABS(VEL)
483 480 CONTINUE
484 C *****
485 E
486 E THE NEXT FEW LINES ARE USED IN FINALIZING ALL THE PLOT PARAMETERS
487 E WHICH INCLUDES AXIS SCALING.
488 E
489 C *****
490 BIGD=BIGDRK
491 IF(BIGDP.GE.BIGDRK) BIGD=BIGDP
492 M=1
493 IF(BIGD.LE.1.0) GO TO 500
494 490 CONTINUE
495 DECD=BIGD/(10.0**M)
496 IF(DECD.LE.1.0) GO TO 520
497 M=M+1
498 GO TO 490
499 500 CONTINUE
500 DIGD=BIGD*(10.0**M)
501 IF(DIGD.GE.1.0) GO TO 510
502 M=M+1
503 GO TO 500
504 510 CONTINUE
505 DECD=DIGD/10.0
506 M=-(M-1)
507 520 CONTINUE
508 IF(DECD.LE.0.1) VFACT=0.1
509 IF((DECD.GT.0.1).AND.(DECD.LE.0.25)) VFACT=0.25
510 IF((DECD.GT.0.25).AND.(DECD.LE.0.5)) VFACT=0.5
511 IF((DECD.GT.0.5).AND.(DECD.LE.1.0)) VFACT=1.0
512 SFACT=VCD/10.0
513 VAD=-VFACT*SFACT*10.0**M
514 DEN=VCD/(2.0*SFACT)
515 VBD=(VFACT/DEN)*10.0**M
516 C
517 M=1
518 IF(TFINAL.LE.1.0) GO TO 540
519 530 CONTINUE
520 DECT=TFINAL/(10.0**M)
521 IF(DECT.LE.1.0) GO TO 560
522 M=M+1
523 GO TO 530
524 540 CONTINUE
525 DIGT=TFINAL*(10.0**M)
526 IF(DIGT.GE.1.0) GO TO 550
527 M=M+1
528 GO TO 540
529 550 CONTINUE
530 DECT=DIGT/10.0
531 M=-(M-1)
532 560 CONTINUE
533 IF(DECT.LE.0.07) HFACT=0.1
534 IF((DECT.GT.0.07).AND.(DECT.LE.0.14)) HFACT=0.2
535 IF((DECT.GT.0.14).AND.(DECT.LE.0.35)) HFACT=0.5
536 IF((DECT.GT.0.35).AND.(DECT.LE.0.7)) HFACT=1.0
537 IF((DECT.GT.0.7).AND.(DECT.LE.1.0)) HFACT=2.0
538 HBD=HFACT*10.0**M
539 LONG=HCD
540 XAD(2)=DFLOAT(LONG)=HBD*0.005
541 C
542 BIGV=BIGVRK
543 IF(BIGVP.GE.BIGVRK) BIGV=BIGVP
544 M=1
545 IF(BIGV.LE.1.0) GO TO 580
546 570 CONTINUE
547 DECV=BIGV/(10.0**M)
548 IF(DECV.LE.1.0) GO TO 600
549 M=M+1
550 GO TO 570

```



```

551      580 CONTINUE
552      DIGV=BIGV*(10.0==M)
553      IF(DIGV.GE.1.0) GO TO 590
554      M=M+1
555      GO TO 580
556      590 CONTINUE
557      DECV=DIGV/10.0
558      M=-(M-1)
559      600 CONTINUE
560      IF(DECV.LE.0.1) VFACT=0.1
561      IF((DECV.GT.0.1).AND.(DECV.LE.0.25)) VFACT=0.25
562      IF((DECV.GT.0.25).AND.(DECV.LE.0.5)) VFACT=0.5
563      IF((DECV.GT.0.5).AND.(DECV.LE.1.0)) VFACT=1.0
564      HAP=-VBD*(VCD/2.0)
565      HBP=VBD
566      HCP=VCD
567      SFACT=VCP/10.0
568      VAP=-VFACT*SFACT*10.0==M
569      DEN=VCP/(2.0*SFACT)
570      VBP=(VFACT/DEN)*10.0==M
571      XAPH(1)=HAP+0.005
572      XAPH(2)=-HAP-0.005
573      YAPH(1)=0.0
574      YAPH(2)=0.0
575      XAPV(1)=0.0
576      XAPV(2)=0.0
577      YAPV(1)=VAP+0.005
578      YAPV(2)=-VAP-0.005
579      C *****
580      C
581      C   THE FOLLOWING SECTION SETS UP THE PIECEWISE LINEAR RESULTS SO THAT
582      C   BETWEEN 50 AND 60 POINTS ARE PLOTTED, DEPENDING ON HOW CLOSE THE
583      C   TOTAL NUMBER OF CALCULATIONS (NPOINT) IS TO AN INTEGRAL MULTIPLE
584      C   OF 50.
585      C
586      C   YP=ARRAY FOR PLOTTED PIECEWISE LINEAR DISPLACEMENTS
587      C   TPP=ARRAY FOR PLOTTED PIECEWISE LINEAR TIME VALUES
588      C   VPP=ARRAY FOR PLOTTED PIECEWISE LINEAR VELOCITIES
589      C
590      C *****
591      INCP=NPOINT/50
592      IDEC=DFLOAT(NPOINT/50)-INCP
593      IF(IDEC.GE.0.5) INCP=NPOINT/50+1
594      NPP=1
595      NDP=1
596      610 CONTINUE
597      YP(NPP)=AMPLP(NDP)
598      TPP(NPP)=TP(NDP)
599      VPP(NPP)=VP(NDP)
600      NPP=NPP+1
601      NDP=NDP+INCP
602      IF(NDP.GE.NPOINT) GO TO 620
603      GO TO 610
604      620 CONTINUE
605      IPOINT=(NDP-NPOINT)-INCP/2
606      IHALF=INCP/2
607      IF(IPOINT.LE.IHALF) GO TO 630
608      YP(NPP)=AMPLP(NDP)
609      TPP(NPP)=TP(NDP)
610      VPP(NPP)=VP(NDP)
611      GO TO 640
612      630 CONTINUE
613      NPP=NPP-1
614      NDP=NDP-INCP
615      640 CONTINUE
616      C *****
617      C
618      C   DUE TO LIMITATIONS IN THE PLOT ROUTINES USED FURTHER ON, THE
619      C   FOLLOWING SETS UP THE RUNGE-KUTTA RESULTS SO THAT EVERY SECOND
620      C   POINT IS PLOTTED IF MORE THAN 1000 POINTS WERE CALCULATED
621      C
622      C *****
623      INCRK=2
624      IF(INTVL.LE.999) INCRK=1
625      NPRK=1
626      NDRK=1
627      650 CONTINUE
628      YRK(NPRK)=AMPLRK(NDRK)
629      TPRK(NPRK)=TRK(NDRK)
630      VPRK(NPRK)=VRK(NDRK)
631      NPRK=NPRK+1
632      NDRK=NDRK+INCRK
633      IF(NDRK.GE.INTVL) GO TO 660
634      GO TO 650
635      660 CONTINUE
636      IF(INCRK.EQ.1) GO TO 670
637      COUNT=DFLOAT(INTVL)
638      INC=COUNT/2.0
639      DELINC=COUNT/2.0-DFLOAT(INC)
640      IF(DELINC.EQ.0) GO TO 680
641      670 CONTINUE
642      YRK(NPRK)=AMPLRK(INTVL)
643      TPRK(NPRK)=TRK(INTVL)
644      VPRK(NPRK)=VRK(INTVL)
645      GO TO 690
646      680 CONTINUE
647      NPRK=NPRK-1
648      NDRK=NDRK-INCRK
649      690 CONTINUE
650      C *****
651      C
652      C   PLOTTING THE SOLUTIONS. CONSULT THE WRITEUP ON CGPL/CGPL2 AND
653      C   THE MANUAL ON DIGITAL PLOTTING FOR DETAILS
654      C
655      C *****
656      ND=NPRK
657      NF=1
658      CALL CGPL2(TPRK,YRK,ND,NF,5,HAD,HBD,HCD,VAD,VBD,VCD,ALPHD)
659      ND=NPP
660      CALL CGPL2(TPP,YP,ND,2,1,HAD,HBD,HCD,VAD,VBD,VCD,ALPHD)

```



```

661      NF=4
662      CALL CGPL2(XAD,YAD,2,NF,4,HAD,HBD,HCD,VAD,VBD,VCD,ALPHD)
663      HA=HAD
664      HC=HCD
665      VA=VAD
666      VC=VCD
667      IPLOT=0
668 700 CONTINUE
669      STARTX=(HC-6.3)/2.0
670      STARTY=VC+0.5
671      HORIZ=HA
672      VERT=VC+0.5
673      CALL PLOT(HORIZ,VERT,3)
674      HORIZ=HC
675      CALL PLOT(HORIZ,VERT,2)
676      VERT=VERT+4.2
677      CALL PLOT(HORIZ,VERT,2)
678      HORIZ=HA
679      CALL PLOT(HORIZ,VERT,2)
680      VERT=VC+0.5
681      CALL PLOT(HORIZ,VERT,2)
682      HORIZ=STARTX+2.0
683      VERT=STARTY+3.8
684      CALL SYMBOL(HORIZ,VERT,0.2,34,0.0,-1)
685      HORIZ=STARTX+2.4
686      CALL SYMBOL(HORIZ,VERT,0.2,'=',0.0,1)
687      HORIZ=STARTX+2.8
688      CALL NUMBER(HORIZ,VERT,0.2,MU,0.0,4)
689      HORIZ=STARTX+0.2
690      VERT=STARTY+3.1
691      CALL SYMBOL(HORIZ,VERT,0.2,52,0.0,-1)
692      HORIZ=STARTX+0.4
693      CALL SYMBOL(HORIZ,VERT,0.2,'T (PIECEWISE ',0.0,12)
694      HORIZ=STARTX+1.0
695      VERT=STARTY+2.8
696      CALL SYMBOL(HORIZ,VERT,0.2,'LINEARIZATION) =' ,0.0,16)
697      HORIZ=STARTX+4.2
698      CALL NUMBER(HORIZ,VERT,0.2,DELTAT,0.0,4)
699      HORIZ=STARTX+0.2
700      VERT=STARTY+2.4
701      CALL SYMBOL(HORIZ,VERT,0.2,52,0.0,-1)
702      HORIZ=STARTX+0.4
703      CALL SYMBOL(HORIZ,VERT,0.2,'T (RUNGE-KUTTA) =' ,0.0,17)
704      HORIZ=STARTX+4.0
705      CALL NUMBER(HORIZ,VERT,0.2,DT,0.0,4)
706      HORIZ=STARTX+0.2
707      VERT=STARTY+1.6
708      CALL SYMBOL(HORIZ,VERT,0.2,'X(O) =' ,0.0,5)
709      HORIZ=STARTX+1.6
710      CALL NUMBER(HORIZ,VERT,0.2,DISPLO,0.0,4)
711      HORIZ=STARTX+3.5
712      CALL SYMBOL(HORIZ,VERT,0.2,'X(O) =' ,0.0,6)
713      HORIZ=STARTX+3.525
714      VERT=STARTY+1.9
715      CALL SYMBOL(HORIZ,VERT,0.2,75,0.0,-1)
716      HORIZ=STARTX+4.9
717      VERT=STARTY+1.6
718      CALL NUMBER(HORIZ,VERT,0.2,VO,0.0,4)
719      HORIZ=STARTX+0.3
720      VERT=STARTY+0.7
721      CALL PLOT(HORIZ,VERT,3)
722      HORIZ=STARTX+0.8
723      CALL PLOT(HORIZ,VERT,2)
724      HORIZ=STARTX+1.0
725      VERT=STARTY+0.6
726      CALL SYMBOL(HORIZ,VERT,0.2,'RUNGE-KUTTA',0.0,11)
727      HORIZ=STARTX+0.6
728      VERT=STARTY+0.3
729      CALL SYMBOL(HORIZ,VERT,0.2,1,0.0,-1)
730      HORIZ=STARTX+1.0
731      VERT=STARTY+0.2
732      CALL SYMBOL(HORIZ,VERT,0.2,'PIECEWISE LINEARIZATION',0.0,23)
733      IF(IPLOT.NE.0) GO TO 710
734      NF=0
735      CALL CGPL2(XAD,YAD,2,NF,4,HAD,HBD,HCD,VAD,VBD,VCD,ALPHD)
736      ND=NPBK
737      NF=1
738      CALL CGPL2(YRK,VPRK,ND,NF,5,HAP,HBP,HCP,VAP,VBP,VCP,ALPHP)
739      ND=NPB
740      CALL CGPL2(YP,VPP,ND,2,1,HAP,HBP,HCP,VAP,VBP,VCP,ALPHP)
741      NF=4
742      CALL CGPL2(XAPH,YAPH,2,NF,4,HAP,HBP,HCP,VAP,VBP,VCP,ALPHP)
743      NF=4
744      CALL CGPL2(XAPV,YAPV,2,NF,4,HAP,HBP,HCP,VAP,VBP,VCP,ALPHP)
745      HA=0.0
746      HC=HCP
747      VA=VAP
748      VC=VCP
749      IPLOT=1
750      GO TO 700
751 710 CONTINUE
752      NF=0
753      CALL CGPL2(XAPV,YAPV,2,NF,4,HAP,HBP,HCP,VAP,VBP,VCP,ALPVP)
754      STOP
755      END
756 C
757 C
758 C
759 C
760 C
761 C
762 C
763 C
764 C
765 C
766 C *****
767 E
768 C THE FOLLOWING SUBROUTINE CALCULATES THE SIGN OF A QUANTITY.
769 C
770 C *****

```



```

771      SUBROUTINE SIG(X,SGN)
772      IMPLICIT REAL*8 (A-H,O-Z)
773      IF(DABS(X).LT.1.0D-08) GO TO 800
774      SGN=X/DABS(X)
775      GO TO 810
776      800 CONTINUE
777      SGN=0.00
778      810 CONTINUE
779      RETURN
780      END
781
782
783
784      C *****
785
786      C      THE FOLLOWING PROGRAM TESTS FOR THE DEGREE OF DAMPING
787      C      (UNDERDAMPING, CRITICAL DAMPING, OR OVERDAMPING) AND THEN
788      C      CALCULATES THE CORRESPONDING DAMPED ANGULAR FREQUENCY.
789      C
790      C *****
791      SUBROUTINE DAMP(N,PSTAR)
792      IMPLICIT REAL*8 (A-H,O-Z)
793      REAL*8 N
794      SO=1.0-N**2
795      CALL SIG(SO,SGNSO)
796      IF(SGNSO.GE.1.0D-08) PSTAR=DSQRT(SO)
797      IF(DABS(SGNSO).LT.1.0D-08) PSTAR=1.0
798      IF(SGNSO.LE.-1.0D-08) PSTAR=DSQRT(-SO)
799      RETURN
800      END
801
802
803
804      C *****
805
806      C      THE FOLLOWING SUBROUTINE CALCULATES THE DISPLACEMENT AT THE END
807      C      OF THE SEGMENT BEING EXAMINED.
808      C
809      C *****
810      SUBROUTINE DISPL(XO,XDOTO,N,DELTAT,P,DELTAX)
811      IMPLICIT REAL*8 (A-H,O-Z)
812      REAL*8 N
813      C1=XO
814      C2=(N*XO+XDOTO)/P
815
816      C *****
817      C      TESTING FOR THE DEGREE OF DAMPING OF THE LINEAR APPROXIMATION. IF
818      C      UNDERDAMPED, THE PROGRAM MOVES TO STEP 900. IF THERE IS CRITICAL
819      C      DAMPING, IT GOES TO STEP 910. FOR OVERDAMPING, THE PROGRAM
820      C      CONTINUES AT STEP 920.
821      C
822      C *****
823      SO=1.0-N**2
824      IF(SO.GE.1.0D-08) GO TO 900
825      IF(DABS(SO).LT.1.0D-08) GO TO 910
826      IF(SO.LE.-1.0D-08) GO TO 920
827      900 CONTINUE
828      DELTAX=DEXP(-N*DELTAT)*(C1+DCOS(P*DELTAT)+C2*DSIN(P*DELTAT))-XO
829      GO TO 930
830      910 CONTINUE
831      DELTAX=DEXP(-P*DELTAT)*(XO+(N*XO+XDOTO)*DELTAT)-XO
832      GO TO 930
833      920 CONTINUE
834      DELTAX=DEXP(-N*DELTAT)*(C1+DCOSH(P*DELTAT)+C2*DSINH(P*DELTAT))-XO
835      930 CONTINUE
836      RETURN
837      END
838
839
840
841      C *****
842
843      C      THE FOLLOWING SUBROUTINE CALCULATES THE VELOCITY AT THE END OF THE
844      C      SEGMENT BEING EXAMINED.
845      C
846      C *****
847      SUBROUTINE SPEED(XO,XDOTO,N,DELTAT,P,XDOT1)
848      IMPLICIT REAL*8 (A-H,O-Z)
849      REAL*8 N
850
851      C *****
852      C      TESTING FOR THE DEGREE OF DAMPING OF THE LINEAR APPROXIMATION. IF
853      C      UNDERDAMPED, THE PROGRAM MOVES TO STEP 1000. IF THERE IS CRITICAL
854      C      DAMPING, IT GOES TO STEP 1010. FOR OVERDAMPING, THE PROGRAM
855      C      CONTINUES AT STEP 1030.
856      C
857      C *****
858      SO=1.0-N**2
859      IF(SO.GE.1.0D-08) GO TO 1000
860      IF(DABS(SO).LT.1.0D-08) GO TO 1010
861      IF(SO.LE.-1.0D-08) GO TO 1020
862      1000 CONTINUE
863      C1=XDOTO
864      C2=-N*(N*XO+XDOTO)/P-P*XO
865      XDOT1=DEXP(-N*DELTAT)*(C1+DCOS(P*DELTAT)+C2*DSIN(P*DELTAT))
866      GO TO 1030
867      1010 CONTINUE
868      XDOT1=DEXP(-P*DELTAT)*(-P*XO+(N*XO+XDOTO)*DELTAT)
869      GO TO 1030
870      1020 CONTINUE
871      C1=XDOTO
872      C2=-N*(N*XO+XDOTO)/P+P*XO
873      XDOT1=DEXP(-N*DELTAT)*(C1+DCOSH(P*DELTAT)+C2*DSINH(P*DELTAT))
874      1030 CONTINUE
875      RETURN
876      END
877
878
879
880      C *****

```



```

881 C
882 C THE FOLLOWING SUBROUTINE CALCULATES THE TIME INTERVAL REQUIRED TO
883 C REACH ZERO VELOCITY (I. E., A PEAK OR TROUGH ON THE DISPLACEMENT-
884 C TIME CURVE) BY USING A NEWTON-RAPHSON METHOD.
885 C
886 C *****
887 SUBROUTINE TIME(XO,XDOTO,DELTA,N,P,DTIME)
888 IMPLICIT REAL*8 (A-H,O-Z)
889 REAL*8 N
890 DELTA=DELTA+0.05
891 C *****
892 C
893 C TESTING FOR THE DEGREE OF DAMPING OF THE LINEAR APPROXIMATION. IF
894 C UNDERDAMPED, THE PROGRAM MOVES TO STEP 1100. IF THERE IS CRITICAL
895 C DAMPING, IT GOES TO STEP 1120. FOR OVERDAMPING, THE PROGRAM
896 C CONTINUES AT STEP 1140.
897 C
898 C *****
899 SQ=1.0-N**2
900 IF(SQ.GE.1.0D-08) GO TO 1100
901 IF(DABS(SQ).LT.1.0D-08) GO TO 1120
902 IF(SQ.LE.-1.0D-08) GO TO 1140
903 C *****
904 C
905 C F=MAIN FUNCTION (VELOCITY)
906 C FP=F-PRIME (ACCELERATION)
907 C
908 C *****
909 1100 CONTINUE
910 C1=XDOTO
911 C2=-N*(N*XO+XDOTO)/P-P*XO
912 C3=-N*C1+P*C2
913 C4=-P*C1-N*C2
914 1110 CONTINUE
915 F=DEXP(-N*DELTA)*(C1*DCOS(P*DELTA)+C2*DSIN(P*DELTA))
916 FP=DEXP(-N*DELTA)*(C3*DCOS(P*DELTA)+C4*DSIN(P*DELTA))
917 DELTA1=DELTA-F/FP
918 DIFF=DELTA1-DELTA
919 IF(DABS(DIFF).LT.1.0D-08) GO TO 1160
920 DELTA=DELTA1
921 GO TO 1110
922 1120 CONTINUE
923 F=-P*XO+(N*XO+XDOTO)*(1.0-P*DELTA)
924 FP=P*(P*XO-(N*XO+XDOTO)*(2.0-P*DELTA))
925 1130 CONTINUE
926 DELTA1=DELTA-F/FP
927 DIFF=DELTA1-DELTA
928 IF(DABS(DIFF).LT.1.0D-08) GO TO 1160
929 DELTA=DELTA1
930 GO TO 1130
931 1140 CONTINUE
932 C1=XDOTO
933 C2=-N*(N*XO+XDOTO)/P+P*XO
934 C3=-N*C1+P*C2
935 C4=P*C1-N*C2
936 1150 CONTINUE
937 F=DEXP(-N*DELTA)*(C1*DCOSH(P*DELTA)+C2*DSINH(P*DELTA))
938 FP=DEXP(-N*DELTA)*(C3*DCOSH(P*DELTA)+C4*DSINH(P*DELTA))
939 DELTA1=DELTA-F/FP
940 DIFF=DELTA1-DELTA
941 IF(DABS(DIFF).LT.1.0D-08) GO TO 1160
942 DELTA=DELTA1
943 GO TO 1150
944 1160 CONTINUE
945 DTIME=DELTA
946 RETURN
947 END
948 C
949 C
950 C
951 C *****
952 C
953 C THE FOLLOWING SUBROUTINE CONTAINS THE DIFFERENTIAL EQUATIONS TO BE
954 C SOLVED BY DVERK: THE INSTANTANEOUS VELOCITY AND THE ORIGINAL
955 C EQUATION.
956 C
957 C *****
958 SUBROUTINE FCH(N,X,Y,YPRIME)
959 IMPLICIT REAL*8 (A-H,O-Z)
960 REAL*8 MU
961 REAL*8 Y(N),YPRIME(N),X
962 COMMON MU
963 YPRIME(1)=Y(2)
964 YPRIME(2)=- (MU*(Y(1)**2-1.0)+YPRIME(1)+Y(1))
965 RETURN
966 END

```

END OF FILE


```

1 C=====
2 C*
3 C*   PROGRAM MATHIEU
4 C*
5 C*
6 C*
7 C*   THE FOLLOWING PROGRAM CALCULATES THE INTEGRAL ORDER SOLUTION
8 C*   FOR THE MATHIEU EQUATION:
9 C*
10 C*    $Y(\text{DOUBLE-PRIME}) + (A-2+0\cdot\cos(2\cdot Z))\cdot Y=0$ 
11 C*
12 C*   OBTAINED BY PIECEWISE LINEARIZATION USING CHORDS.
13 C*
14 C*
15 C*   THE RESULTS ARE BASED ON THE STANDARD FORM:
16 C*
17 C*    $Y=\text{ALPHA}\cdot\cos(P\cdot Z)+\text{BETA}\cdot\sin(P\cdot Z)$ 
18 C*
19 C*    $P=\text{SORT}(A-2+0\cdot\cos(2\cdot Z))$ 
20 C*
21 C*
22 C*   THE CHARACTERISTIC NUMBER, A, IS FOUND BY MAKING AN INITIAL
23 C*   GUESS AND FINDING THE CLOSEST VALUE (EITHER ODD OR EVEN
24 C*   ORDER) USING A METHOD OF CONTINUED FRACTIONS, DESCRIBED
25 C*   FURTHER ON.
26 C*
27 C*
28 C*   THE PROGRAM ALSO CALCULATES THE APPROPRIATE MATHIEU FUNCTIONS
29 C*   USING A SERIES APPROXIMATION, AS WELL AS A RUNGE-KUTTA
30 C*   SOLUTION USING FIFTH- AND SIXTH-ORDER METHODS. ALL THREE
31 C*   RESULTS ARE PLOTTED.
32 C*
33 C=====
34 C
35 C
36 C
37 C
38 C
39 C
40 C
41 C
42 C
43 C
44 C   IMPLICIT REAL*8 (A-H,O-Z)
45 C   EXTERNAL FCN
46 C   REAL*8 Y(2),C(24),W(2,20)
47 C   REAL*8 LOW,LOWOUT,MID,MIDOUT
48 C   REAL*8 RATIO1/1000,RATIO2(100),COEFF(100),T(1000)
49 C   REAL*4 AMPLP(1000),AMPLS(1000),AMPLRK(1000),Z(1000)
50 C   REAL*4 YP(1000),ZP(1000),YRK(1000),ZPRK(1000)
51 C   REAL*4 HA,HB,HC,VA,VB,VC
52 C   REAL*4 XA(2),YA(2)
53 C   INTEGER*4 ALPH(20)
54 C   COMMON/CALC/AIN,DISPLO,DIFF,AOUT,IPAR
55 C   COMMON/CHAR/A
56 C   COMMON/PARAM/O
57 C=====
58 C
59 C   THE FOLLOWING VALUES ARE READ FROM DATA FILE MATHIEUDATA:
60 C
61 C
62 C   AGUESS=INITIAL GUESS FOR CHARACTERISTIC NUMBER
63 C   O=SAME AS IN EQUATION
64 C
65 C   DISPLO=INITIAL AMPLITUDE
66 C   VO=INITIAL VELOCITY
67 C   TO=INITIAL TIME
68 C
69 C   DELTA=INCREMENT/DECREMENT FOR A (USED IN SCAN PHASE)
70 C   H=NUMBER OF DIVISIONS OF PI (AN INTEGRAL MULTIPLE OF 50.0, USED
71 C   FOR FINDING INCREMENT OF PI)
72 C
73 C   HA,HC=PLOT PARAMETERS FOR HORIZONTAL AXIS (HC AN INTEGRAL MULTIPLE
74 C   OF 10.0)
75 C
76 C   VC=PLOT PARAMETER FOR VERTICAL AXIS (AN INTEGRAL MULTIPLE OF 12.0
77 C   GIVING BEST RESULTS)
78 C
79 C   XA,YA=PARAMETERS FOR PLOTTING ZERO LINE
80 C
81 C   ALPH=ARRAY FOR PLOT AXIS LABELS
82 C
83 C=====
84 C   READ(5,10) AGUESS,O
85 C   10 FORMAT(2D10.6)
86 C   READ(5,20) DISPLO,VO,ZO
87 C   20 FORMAT(3D10.6)
88 C   READ(5,30) DELTA,H
89 C   30 FORMAT(2D10.4)
90 C   READ(5,40) HA,HC
91 C   40 FORMAT(2F10.4)
92 C   READ(5,50) VC
93 C   50 FORMAT(F10.4)
94 C   READ(5,60) XA(1)
95 C   60 FORMAT(F10.4)
96 C   READ(5,70) YA(1),YA(2)
97 C   70 FORMAT(2F10.4)
98 C   READ(5,80) (ALPH(I),I=1,12)
99 C   80 FORMAT(12A4)
100 C   READ(5,90) (ALPH(I),I=13,16)
101 C   90 FORMAT(4A4)
102 C   READ(5,100) (ALPH(I),I=17,20)
103 C   100 FORMAT(4A4)
104 C=====
105 C
106 C   DELTAZ=INCREMENT OF Z (SINCE SOLUTIONS ARE PERIODIC IN EITHER PI
107 C   OR 2*PI)
108 C
109 C=====
110 C   PI=3.141592653589793

```



```

111      DELTA2=PI/H
112      C*****
113      C
114      C      THE FOLLOWING ARE USED AS PROGRAM FLAGS FOR CALCULATING EITHER THE
115      C      ODD OR EVEN INTEGRAL ORDER CHARACTERISTIC NUMBERS. THE ODD ONE IS
116      C      FOUND FIRST AND THEN THE EVEN
117      C
118      C      IPAR=FLAG FOR DETERMINING ORDER ("0" INDICATES ODD, "1" EVEN)
119      C      I=COUNTER FOR DETERMINING HOW MANY TIMES THE SET OF CALCULATIONS
120      C      HAVE BEEN DONE ("1" MEANS THE ODD ORDER VALUE IS BEING FOUND,
121      C      "2" THE EVEN)
122      C
123      C*****
124      C      IPAR=0
125      C      I=1
126      C
127      C*****
128      C      THE EQUATION PARAMETERS ARE WRITTEN INTO OUTPUT FILE MATHIEUNUM
129      C
130      C*****
131      C      WRITE(10,110)
132      C      110 FORMAT('1','RESULTS FOR THE EQUATION:')
133      C      WRITE(10,120)
134      C      120 FORMAT('0','Y(DOUBLE-PRIME)+(A-2*Q*COS(2*Z))=Y=0')
135      C      WRITE(10,130) DISPL0,V0
136      C      130 FORMAT('0','Y(0)=',1X,G20.10,2X,'Y(PRIME)(0)=',1X,G20.10)
137      C      WRITE(10,140) AGUESS,0
138      C      140 FORMAT('0','A(GUESS)=',1X,G20.10,2X,'Q=',1X,G20.10)
139      C*****
140      C
141      C      THE FOLLOWING SECTION (ENDING AT STEP 450) CALCULATES THE
142      C      CHARACTERISTIC NUMBER FOR A GIVEN AGUESS AND Q, BASED UPON THE
143      C      INITIAL DISPLACEMENT AND THE PROXIMITY OF AGUESS TO THE
144      C      APPROPRIATE ODD AND EVEN ORDER RESULTS.
145      C
146      C      INITIALLY, A RANGE OF VALUES BETWEEN WHICH THE FINAL RESULT FOR
147      C      THE PARTICULAR ORDER IN QUESTION LIES IS FOUND (STEPS 160 TO 300)
148      C      THIS IS DONE BY EITHER INCREMENTING OR DECREMENTING AGUESS BY A
149      C      FIXED AMOUNT UNTIL THE LIMITS ARE FOUND, WITH THE DIRECTION OF THE
150      C      SEARCH (EITHER ABOVE OR BELOW AGUESS) DEPENDING UPON THE OUTPUT
151      C      VALUE FROM A SET OF RECURRENCE RELATIONS USING AGUESS AS THE
152      C      INPUT.
153      C
154      C      ONCE THE BOUNDS ARE KNOWN, THE PROGRAM FOCUSES TO THE FINAL RESULT
155      C      BY INTERVAL HALVING (STEPS 300 TO 350). HERE THE DIRECTION IS
156      C      DETERMINED BY THE OUTPUT VALUE FROM THE SAME RECURRENCE RELATIONS
157      C      USING THE FIRST MIDPOINT AS THE INITIAL INPUT
158      C
159      C      ONCE THE CHARACTERISTIC NUMBERS FOR THE ODD AND EVEN ORDERS HAVE
160      C      BEEN FOUND, THE ONE CLOSEST TO AGUESS IS THE VALUE THAT IS USED
161      C      IN CALCULATING THE SOLUTION.
162      C
163      C
164      C
165      C      AIN=INPUT VALUE FOR RECURRENCE RELATIONS
166      C      RECUR=SUBROUTINE FOR RECURRENCE RELATIONS
167      C      DIFF=DIFFERENCE BETWEEN AIN AND OUTPUT VALUE
168      C
169      C
170      C
171      C**NOTE**IF AT ANY POINT THE INPUT AND OUTPUT VALUES ARE NEARLY EQUAL,
172      C      THE SEARCH FOR THE UPPER AND LOWER LIMITS OR THE FOCUSING PHASE
173      C      STOPS, THE VALUE FOR IPAR IS FOUND (WHICH DETERMINES THE
174      C      PERIOD), AND THE OUTPUT VALUE TAKEN AS THE FINAL RESULT.
175      C
176      C*****
177      C      150 CONTINUE
178      C      AIN=AGUESS
179      C      CALL RECUR
180      C      IF(DABS(DIFF).LE.1.0D-08) GO TO 410
181      C*****
182      C
183      C      THE FOLLOWING CONSIDERS A POSITIVE INPUT VALUE FOR THE
184      C      CHARACTERISTIC NUMBER (STEPS 160 TO 230)
185      C
186      C      ADUT=OUTPUT VALUE FROM RECUR
187      C      UP=UPPER LIMIT
188      C      LOW=LOWER LIMIT
189      C
190      C
191      C      1 ADUT POSITIVE
192      C
193      C      IF AIN IS GREATER THAN ADUT (SEE STEP 180), AIN IS DECREMENTED AND
194      C      A NEW ADUT CALCULATED (SHOULD AIN BECOME LESS THAN ZERO, THE
195      C      PROGRAM CONTINUES AT STEP 230). IF AIN IS GREATER THAN ADUT, THE
196      C      PROGRAM GOES TO STEP 190. IF ADUT IS POSITIVE, THE PROGRAM
197      C      RETURNS TO STEP 180. IF AIN IS LESS THAN ADUT OR IF AIN IS
198      C      GREATER THAN ADUT (ADUT BEING NEGATIVE), STEP 200 IS NEXT. THE
199      C      LOWER BOUND IS THE CURRENT AIN, THE UPPER LIMIT THE PREVIOUS ONE,
200      C      AND THE PROGRAM MOVES TO STEP 300
201      C
202      C      IF AIN IS LESS THAN ADUT, AIN IS INCREMENTED AND A NEW ADUT FOUND
203      C      UNTIL AIN IS GREATER THAN ADUT (STARTING AT STEP 210, RETURNING
204      C      TO STEP 210 IF NOT). THE UPPER BOUND IS THE CURRENT AIN AND THE
205      C      LOWER THE PREVIOUS ONE. THE PROGRAM THEN MOVES TO STEP 300 TO
206      C      BEGIN THE FINAL SET OF ITERATIONS
207      C
208      C
209      C      2. ADUT NEGATIVE
210      C
211      C      AIN IS DECREMENTED (STARTING AT STEP 220) AND A NEW ADUT IS FOUND
212      C      UNTIL ADUT BECOMES POSITIVE, THE PROGRAM CONTINUING AT STEP 170
213      C      IF ADUT IS STILL NEGATIVE, IT GOES BACK TO STEP 220. IF AIN GETS
214      C      LESS THAN ZERO, THE PROGRAM GOES TO STEP 230
215      C
216      C*****
217      C      IF(AIN.LE.0.D0) GO TO 230
218      C      160 CONTINUE
219      C      IF(ADUT.LE.0.D0) GO TO 220
220      C      170 CONTINUE

```



```

221      IF(AIN.LE.AOUT) GO TO 210
222  C
223      180 CONTINUE
224      AIN=AIN-DELTA
225      IF(AIN.LE.O.DO) GO TO 230
226      CALL RECUR
227      IF(DABS(DIFF).LE.1.0D-08) GO TO 410
228      IF(AIN.GT.AOUT) GO TO 190
229      GO TO 200
230      190 CONTINUE
231      IF(AOUT.GT.O.DO) GO TO 180
232      200 CONTINUE
233      UP=AIN+DELTA
234      LOW=AIN
235      GO TO 300
236  C
237      210 CONTINUE
238      AIN=AIN+DELTA
239      CALL RECUR
240      IF(DABS(DIFF).LE.1.0D-08) GO TO 410
241      IF(AIN.LE.AOUT) GO TO 210
242      UP=AIN
243      LOW=AIN-DELTA
244      GO TO 300
245  C
246      220 CONTINUE
247      AIN=AIN-DELTA
248      IF(AIN.LE.O.DO) GO TO 230
249      CALL RECUR
250      IF(DABS(DIFF).LE.1.0D-08) GO TO 410
251      IF(AOUT.GT.O.DO) GO TO 170
252      GO TO 220
253  C *****
254  C
255      THE FOLLOWING APPLIES FOR A NEGATIVE AIN (STEPS 230 TO 300).
256  C
257  C
258      1. AOUT NEGATIVE
259  C
260      IF AIN IS LESS THAN AOUT (STARTING AT SETP 250), AIN IS
261      INCREMENTED AND A NEW AOUT FOUND, THE PROGRAM GOING TO STEP 180
262      FOR A POSITIVE VALUE OF AIN. IF AIN IS LESS THAN AOUT, THE
263      PROGRAM MOVES TO STEP 280. IF AOUT IS STILL NEGATIVE, THE NEXT
264      IS TO STEP 270. THE UPPER LIMIT IS THE CURRENT AIN AND THE
265      LOWER THE PREVIOUS ONE. THE FINAL STAGE THEN STARTS AT STEP
266      300.
267  C
268      IF AIN IS GREATER THAN AOUT, AIN IS DECREMENTED (STARTING AT STEP
269      280) AND A NEW AOUT FOUND UNTIL AIN IS LESS THAN AOUT, THE PROGRAM
270      RETURNING TO STEP 280 IF NOT. THE UPPER LIMIT IS THE PREVIOUS AIN
271      AND THE LOWER THE CURRENT ONE. STEP 300 IS NEXT.
272  C
273  C
274      2. AOUT POSITIVE
275  C
276      STARTING AT STEP 290, AIN IS INCREMENTED AND A NDW AOUT IS FOUND
277      UNTIL AOUT IS NEGATIVE, THEN MOVING TO STEP 240 AND CONTINUING
278      FROM THERE. IF AOUT REMAINS POSITIVE, THE PROGRAM RETURNS TO STEP
279      290. SHOULD AIN BECOME POSITIVE, THE PROGRAM MOVES TO STEP 280.
280  C
281  C
282  C *****
283      230 CONTINUE
284      IF(AOUT.GT.O.DO) GO TO 290
285      240 CONTINUE
286      IF(AIN.GT.AOUT) GO TO 280
287  C
288      250 CONTINUE
289      AIN=AIN+DELTA
290      IF(AIN.GT.O.DO) GO TO 160
291      CALL RECUR
292      IF(DABS(DIFF).LE.1.0D-08) GO TO 410
293      IF(AIN.LE.AOUT) GO TO 260
294      GO TO 270
295      260 CONTINUE
296      IF(AOUT.LE.O.DO) GO TO 250
297      270 CONTINUE
298      UP=AIN
299      LOW=AIN-DELTA
300      GO TO 300
301  C
302      280 CONTINUE
303      AIN=AIN-DELTA
304      CALL RECUR
305      IF(DABS(DIFF).LE.1.0D-08) GO TO 410
306      IF(AIN.GT.AOUT) GO TO 280
307      UP=AIN+DELTA
308      LOW=AIN
309      GO TO 300
310  C
311      290 CONTINUE
312      AIN=AIN+DELTA
313      IF(AIN.GT.O.DO) GO TO 160
314      CALL RECUR
315      IF(DABS(DIFF).LE.1.0D-08) GO TO 410
316      IF(AOUT.LE.O.DO) GO TO 240
317      GO TO 290
318  C *****
319  C
320      THE FOLLOWING SECTION IS THE FINAL SET OF ITERATIONS TO FIND THE
321      CHARACTERISTIC NUMBER BY USING INTERVAL HALVING.
322  C
323      MID=INTERVAL MIDPOINT (AVERAGE OF UP AND LOW)
324      MIDOUT=OUTPUT FROM RECUR USING MID AS INPUT
325  C
326      BASICALLY, IF MID IS GREATER THAN MIDOUT, MID BECOMES THE NEW
327      UPPER LIMIT, AND IF LESS THAN MIDOUT, THE NEW LOWER BOUND
328  C
329  C *****
330      300 CONTINUE

```



```

331      MID=(UP+LOW)/2.0
332      AIN=MID
333      CALL RECUR
334      IF(DABS(DIFF).LE.1.0D-08) GO TO 350
335      MIDOUT=AOUT
336      IF(MID.LE.0.00) GO TO 330
337      IF(MIDOUT.LE.0.00) GO TO 320
338      IF(MID.LE.MIDOUT) GO TO 320
339  310 CONTINUE
340      UP=MID
341      GO TO 300
342  320 CONTINUE
343      LOW=MID
344      GO TO 300
345  330 CONTINUE
346      IF(MIDOUT.LE.0.00) GO TO 340
347      GO TO 320
348  340 CONTINUE
349      IF(MID.LE.MIDOUT) GO TO 320
350      GO TO 310
351  C *****
352  C
353  C      FROM STEPS 350 TO 390, THE PROGRAM DETERMINES IF BOTH THE ODD AND
354  C      EVEN ORDER SOLUTIONS HAVE BEEN CALCULATED, AND THEN WRITING OUT
355  C      THE APPROPRIATE VALUES INTO OUTPUT FILE MATHIEUNUM ONCE THIS HAS
356  C      BEEN DONE.
357  C
358  C      AODD=ODD ORDER CHARACTERISTIC NUMBER
359  C      AEVEN=EVEN ORDER NUMBER
360  C
361  C *****
362  350 CONTINUE
363      IF(I.NE.1) GO TO 360
364      AODD=AOUT
365      IPAR=1
366      I=2
367      GO TO 150
368  360 CONTINUE
369      AEVEN=AOUT
370      WRITE(10,370) AODD,AEVEN
371  370 FORMAT('0','A(ODD)=',1X,G20.10,2X,'A(EVEN)=',1X,G20.10)
372  C *****
373  C
374  C      HERE THE PROGRAM DETERMINES WHICH CHARACTERISTIC NUMBER THAT
375  C      AGUESS IS CLOSER TO, AODD OR AEVEN, AND THEN WRITING INTO
376  C      MATHIEUNUM THE RESULT, USING THIS VALUE FOR CALCULATING THE
377  C      SOLUTIONS.
378  C
379  C      FACT=FACTOR FOR DETERMINING SOLUTION PERIOD (1.0 IS FOR EVEN ORDER
380  C      AND 2.0 FOR ODD)
381  C
382  C *****
383      IF(AODD.LE.AEVEN) GO TO 390
384      IF(AGUESS.LE.AODD) GO TO 380
385      GO TO 440
386  380 CONTINUE
387      IF(AGUESS.LE.AEVEN) GO TO 420
388      DIFF1=AODD-AGUESS
389      DIFF2=AGUESS-AEVEN
390      IF(DIFF1.LE.DIFF2) GO TO 440
391      GO TO 420
392  C
393  390 CONTINUE
394      IF(AGUESS.LE.AEVEN) GO TO 400
395      GO TO 420
396  400 CONTINUE
397      IF(AGUESS.LE.AODD) GO TO 440
398      DIFF1=AEVEN-AGUESS
399      DIFF2=AGUESS-AODD
400      IF(DIFF1.GT.DIFF2) GO TO 440
401      GO TO 420
402  410 CONTINUE
403      IF(IPAR.EQ.1) GO TO 440
404  C
405  420 CONTINUE
406      A=AEVEN
407      IPAR=1
408      FACT=1.0
409      WRITE(10,430) A
410  430 FORMAT('0','CHARACTERISTIC NUMBER (EVEN ORDER)=',1X,G20.10)
411      GO TO 460
412  440 CONTINUE
413      A=AODD
414      FACT=2.0
415      IPAR=0
416      WRITE(10,450) A
417  450 FORMAT('0','CHARACTERISTIC NUMBER (ODD ORDER)=',1X,G20.10)
418  C *****
419  C
420  C      STEPS 460 TO 500 ARE THE PIECEWISE LINEAR SOLUTION.
421  C
422  C      AMPLP=ARRAY FOR PIECEWISE LINEAR DISPLACEMENT
423  C      BIGP=LARGEST VALUE FOR AMPLP (TO BE USED FOR PLOT SCALING)
424  C      Z=ARRAY FOR INSTANTANEOUS NORMALIZED VALUES FOR Z
425  C      XO=INITIAL DISPLACEMENT OF LINEAR SEGMENT
426  C      XPO=INITIAL VELOCITY OF LINE
427  C      NPOINT=NUMBER OF DATA POINTS (USED FOR PLOTTING)
428  C
429  C *****
430  460 CONTINUE
431      AMPLP(1)=DISPLO
432      BIGP=DABS(DISPLO)
433      Z(1)=ZO/PI
434      T(1)=ZO
435      XO=DISPLO
436      XPO=VO
437      NPOINT=FACT*H
438  C *****
439  C
440  C      IK=COUNTER FOR FILLING REMAINING ELEMENTS OF AMPLP AND Z

```



```

441 C      PSO=SQUARE OF ANGULAR FREQUENCY OF APPROXIMATED
442 C      X=FINAL DISPLACEMENT OF LINEAR SEGMENT
443 C      XP=FINAL LINE VELOCITY
444 C
445 C
446 C      IF PSO IS POSITIVE, THE SOLUTION IS OF THE FAMILIAR TRIGONOMETRIC
447 C      FORM. IF ZERO, THE SOLUTION IS LINEAR (STEP 470) AND IF NEGATIVE,
448 C      HYPERBOLIC (STEP 480).
449 E
450 C *****
451 DO 500 I=1,NPOINT
452   IP=I+1
453   PSO=A-2.0*O*DCOS((2.0*DFLOAT(I)-1.0)*DELTAZ)
454   IF(PSO.LT.-1.0D-10) GO TO 480
455   IF(DABS(PSO).LE.1.0D-10) GO TO 470
456   P=DSORT(PSO)
457   X=XO*DCOS(P*DELTAZ)+(XPO/P)*DSIN(P*DELTAZ)
458   XP=-P*XO*DSIN(P*DELTAZ)+XPO*DCOS(P*DELTAZ)
459   GO TO 490
460 470 CONTINUE
461   X=XPO*DELTAZ+XO
462   XP=XPO
463   GO TO 490
464 480 CONTINUE
465   P=DSORT(-PSO)
466   X=XO*DCOSH(P*DELTAZ)+(XPO/P)*DSINH(P*DELTAZ)
467   XP=P*XO*DSINH(P*DELTAZ)+XPO*DCOSH(P*DELTAZ)
468 490 CONTINUE
469   AMPLP(IP)=X
470   Z(IP)=DFLOAT(I)*DELTAZ/PI
471 C *****
472 C
473 C      T=DOUBLE-PRECISION ARRAY FOR Z (TO BE USED FOR FINDING SERIES
474 C      APPROXIMATION)
475 E
476 C *****
477 T(IP)=DFLOAT(I)*DELTAZ
478 XO=X
479 XPO=XP
480 IF(DABS(X).GT.BIGP) BIGP=DABS(X)
481 500 CONTINUE
482 C *****
483 C
484 C      THE FOLLOWING CALCULATES THE COEFFICIENTS OF THE SERIES
485 C      APPROXIMATION. USING THE METHOD DESCRIBED BY E.L. INCE IN
486 C      "TABLES OF THE ELLIPTIC CYLINDER FUNCTIONS" (PROC. ROY. SOC.
487 C      EDINBURGH, VOL. 52, P. 355-423, 1932) AND N.W. MCLACHLAN IN
488 C      "THEORY AND APPLICATION OF MATHIEU FUNCTIONS", CH. 3, P. 28 FF.,
489 C      OXFORD UNIVERSITY PRESS, 1947
490 C
491 C      IR1=COUNTER FOR CALCULATING PRELIMINARY COEFFICIENT RATIOS
492 C      GAMMA=FACTOR USED FOR CALCULATING CONTINUED FRACTIONS USED FOR
493 C      ABOVE
494 C
495 C      FOR A NON-ZERO INITIAL DISPLACEMENT AND AN EVEN ORDER
496 C      CHARACTERISTIC NUMBER, THE PROGRAM MOVES TO STEP 510. FOR A ZERO
497 C      INITIAL DISPLACEMENT AND EVEN ORDER, THE CALCULATIONS START AT
498 C      STEP 570. FOR NON-ZERO INITIAL DISPLACEMENT AND ODD ORDER, THE
499 C      ITERATIONS BEGIN AT STEP 580. FOR ODD ORDER AND ZERO
500 C      DISPLACEMENT, STEP 600 IS WHERE THE PROGRAM CONTINUES.
501 E
502 C *****
503 IR1=24
504 GAMMA=0.0
505 IF((DISPLO.NE.O.O).AND.(IPAR.EQ.1)) GO TO 510
506 IF((DISPLO.EQ.O.O).AND.(IPAR.EQ.1)) GO TO 570
507 IF((DISPLO.NE.O.O).AND.(IPAR.EQ.O)) GO TO 580
508 GO TO 600
509 C *****
510 E
511 E
512 C      THE CASE OF NON-ZERO INITIAL DISPLACEMENT AND EVEN ORDER.
513 C
514 C      RATIO=RATIO BETWEEN COEFFICIENT FOR SPECIFIC TERM AND ONE FOR
515 C      PREVIOUS TERM
516 C      SUM=TERM USED IN SUMMING SQUARES OF VARIOUS RATIO VALUES
517 C      IEND=COUNTER USED IN CALCULATING VARIOUS VALUES FOR RATIO
518 C *****
519 510 CONTINUE
520 RATIO1(1)=A/O
521 RATIO1(2)=(A-4.0)/Q-2.0/RATIO1(1)
522 SUM=2.0
523 IEND=2
524 C *****
525 C
526 C      THE CONTINUED FRACTION FOR CALCULATING RATIO.
527 C
528 C      I1STAR=COUNTER USED FOR RATIO1
529 C *****
530 520 CONTINUE
531 I1STAR=IR1
532 IF((DISPLO.EQ.O.O).AND.(IPAR.EQ.1)) I1STAR=IR1-1
533 RATIO1(I1STAR)=O/(A-(2.0*DFLOAT(IR1))*2-Q+GAMMA)
534 GAMMA=RATIO1(I1STAR)
535 IR1=IR1-1
536 IF(IR1.EQ.IEND) GO TO 530
537 GO TO 520
538 C *****
539 C
540 C      RATIO2=RATIO BETWEEN COEFFICIENT FOR SPECIFIC TERM AND COEFFICIENT
541 C      FOR PREVIOUS ONE
542 C      IR2,I2STAR=COUNTERS FOR RATIO2
543 C *****
544 C
545 C *****
546 530 CONTINUE
547 RATIO2(1)=RATIO1(1)
548 DD 540 IR2=1,23
549 I2STAR=IR2+1
550 RATIO2(I2STAR)=RATIO2(IR2)*RATIO1(I2STAR)

```



```

551      540 CONTINUE
552      C *****
553      C
554      C      BY SUMMING ALL THE VARIOUS VALUES FOR RATIO2, AND INVERTING THE
555      C      SQUARE ROOT, THE VALUE FOR THE FIRST COEFFICIENT IS DETERMINED AND
556      C      SUBSEQUENTLY, ALL THE REMAINING ONES.
557      C
558      C      COEFF=ARRAY FOR COEFFICIENT VALUES
559      C      IC1,IC2=COUNTERS FOR COEFF
560      C
561      C *****
562      C      IR1=24
563      C      DO 550 IS=1,IR1
564      C      SUM=SUM+RATIO2(IS)**2
565      550 CONTINUE
566      C      COEFF(1)=1.0/DSQRT(SUM)
567      C      DO 560 IC1=1,IR1
568      C      IC2=IC1+1
569      C      COEFF(IC2)=COEFF(1)*RATIO2(IC1)
570      560 CONTINUE
571      C      GO TO 610
572      C *****
573      C
574      C      INITIALIZING THE CALCULATIONS FOR THE SITUATION OF ZERO INITIAL
575      C      AND EVEN ORDER.
576      C
577      C *****
578      570 CONTINUE
579      C      RATIO1(1)=(A-4.0)/Q
580      C      SUM=1.0
581      C      IEND=1
582      C      IR1=25
583      C      GO TO 520
584      C *****
585      C
586      C      SETTING UP THE CASE OF NON-ZERO DISPLACEMENT AND ODD ORDER.
587      C
588      C *****
589      580 CONTINUE
590      C      RATIO1(1)=(A-1.0-Q)/Q
591      C      SUM=1.0
592      C      IEND=1
593      590 CONTINUE
594      C      RATIO1(IR1)=Q/(A-(2.0*DFLOAT(IR1)+1.0)**2-Q=GAMMA)
595      C      GAMMA=RATIO1(IR1)
596      C      IR1=IR1-1
597      C      IF(IR1.EQ.IEND) GO TO 530
598      C      GO TO 590
599      C *****
600      C
601      C      THE SITUATION OF ZERO INITIAL DISPLACEMENT AND ODD ORDER.
602      C
603      C *****
604      600 CONTINUE
605      C      RATIO1(1)=(A-1.0+Q)/Q
606      C      SUM=1.0
607      C      IEND=1
608      C      GO TO 590
609      C *****
610      C
611      C      THE SECTION FROM STEP 610 TO STEP 650 IS THE SERIES APPROXIMATION
612      C      THAT IS DESCRIBED IN INCE AND MCLACHLAN.
613      C
614      C      BIGS=LARGEST VALUE FOR SERIES AMPLITUDE (FOR PLOT SCALING)
615      C
616      C *****
617      610 CONTINUE
618      C      BIGS=DABS(DISPLO)
619      C      NPOINT=FACT*H+1.0
620      C *****
621      C
622      C      INITIALIZING THE CALCULATIONS OF THE SERIES TERMS.
623      C
624      C      DENOM=SUMMATION OF SERIES COEFFICIENTS (FOR NON-ZERO INITIAL
625      C      DISPLACEMENTS) OR COEFFICIENTS AND TERM NUMBER (FOR ZERO
626      C      INITIAL DISPLACEMENTS) USED IN DETERMINING MATHIEU FUNCTION
627      C      COEFFICIENT FOR COMPLETE SOLUTION
628      C      TERM=SUM OF INDIVIDUAL TERMS FOR MATHIEU FUNCTION APPROXIMATION
629      C
630      C *****
631      C      DO 650 J=1,NPOINT
632      C      DENOM=0.0
633      C      TERM=0.0
634      C      IF(DISPLO.EQ.0.0) GO TO 630
635      C *****
636      C
637      C      RESULTS FOR NON-ZERO INITIAL DISPLACEMENT
638      C
639      C      THETA=TERM NUMBER
640      C
641      C *****
642      C      DO 620 K=1,25
643      C      IF(IPAR.EQ.0) THETA=2.0*DFLOAT(K)-1.0
644      C      IF(IPAR.EQ.1) THETA=2.0*DFLOAT(K)-2.0
645      C      DENOM=DENOM+COEFF(K)
646      C      TERM=TERM+COEFF(K)*DCOS(THETA*T(J))
647      620 CONTINUE
648      C *****
649      C
650      C      AMPLS=ARRAY FOR INSTANTANEOUS AMPLITUDE FOR SERIES SOLUTION
651      C      DISPLS=DOUBLE-PRECISION VALUE OF AMPLS
652      C
653      C *****
654      C      AMPLS(J)=DISPLO*TERM/DENOM
655      C      DISPLS=DBLE(AMPLS(J))
656      C      IF(DABS(DISPLS).GT.BIGS) BIGS=DABS(DISPLS)
657      C      GO TO 650
658      630 CONTINUE
659      C *****
660      C

```



```

661 C      CALCULATIONS FOR ZERO INITIAL DISPLACEMENT.
662 C
663 C *****
664 DO 640 K=1,25
665 IF (IPAR.EQ.0) THETA=2.0*DFLOAT(K)-1.0
666 IF (IPAR.EQ.1) THETA=2.0*FLOAT(K)
667 DENOM=DENOM+THETA*COEFF(K)
668 TERM=TERM+COEFF(K)*DSIN(THETA*T(J))
669 640 CONTINUE
670 AMPLS(J)=VO/TERM/DENOM
671 DISPLS=DSLE(AMPLS(J))
672 IF (DABS(DISPLS).GT.BIGS) BIGS=DABS(DISPLS)
673 650 CONTINUE
674 C *****
675 C
676 THE SECTION ENDING AT STEP 660 IS THE RUNGE-KUTTA SOLUTION.
677 C
678 C
679 C
680 THE FOLLOWING ARE PARAMETERS FOR THE RUNGE-KUTTA ROUTINE.
681 CONSULT THE IMSL MANUAL ON "DVERK" FOR FURTHER DETAILS.
682 C
683 C *****
684 NPOINT=FACT*H+1.0
685 BIGRK=DABS(DISPLS)
686 Y(1)=DISPLS
687 Y(2)=VO
688 ZRK=ZO
689 NEON=2
690 NW=2
691 TOL=1.0D-12
692 IND=1
693 ZEND=DELTAZ
694 DO 660 K=1,NPOINT
695 AMPLRK(K)=Y(1)
696 CALL DVERK(NEON,FCN,ZRK,Y,ZEND,TOL,IND,C,NW,W,IER)
697 C *****
698 C
699 INCREMENTING ZEND.
700 C
701 C *****
702 ZEND=ZEND+DELTAZ
703 IF (DABS(Y(1)).GT.BIGRK) BIGRK=Y(1)
704 660 CONTINUE
705 C *****
706 C
707 WRITING THE SOLUTIONS INTO OUTPUT FILE MATHIEUNUM.
708 C
709 C *****
710 WRITE(10,670)
711 670 FORMAT('O',11X,'Z',13X,'PIECEWISE LINEARIZATION',2X,'SERIES APPROX
712 IMATION',8X,'RUNGE-KUTTA')
713 DO 690 L=1,NPOINT
714 WRITE(10,680) Z(L),AMPL(L),AMPLS(L),AMPLRK(L)
715 680 FORMAT(1X,4(G20.10,4X))
716 690 CONTINUE
717 C *****
718 C
719 THE NEXT FEW LINES ARE USED IN FINALIZING ALL THE PLOT PARAMETERS
720 WHICH INCLUDES AXIS SCALING.
721 C
722 C *****
723 INC=HC
724 XA(2)=Z(NPOINT)-0.005
725 BIG=BIGRK
726 IF ((BIGP.GT.BIGS).AND.(BIGP.GT.BIGRK)) BIG=BIGP
727 IF ((BIGS.GT.BIGP).AND.(BIGS.GT.BIGRK)) BIG=BIGS
728 M=1
729 IF (BIG.LE.1.0) GO TO 710
730 CONTINUE
731 DEC=BIG/(10.0**M)
732 IF (DEC.LE.1.0) GO TO 730
733 M=M+1
734 GO TO 700
735 710 CONTINUE
736 DIG=BIG*(10.0**M)
737 IF (DIG.GE.1.0) GO TO 720
738 M=M+1
739 GO TO 710
740 720 CONTINUE
741 DEC=DIG/10.0
742 M=(M-1)
743 730 CONTINUE
744 IF (DEC.LE.0.125) VFACT=0.125
745 IF ((DEC.GT.0.125).AND.(DEC.LE.0.25)) VFACT=0.25
746 IF ((DEC.GT.0.25).AND.(DEC.LE.0.5)) VFACT=0.5
747 IF ((DEC.GT.0.5).AND.(DEC.LE.1.0)) VFACT=1.0
748 SFACT=VC/10.0
749 VA=-VFACT*SFACT*10.0**M
750 DEN=VC/(2.0*SFACT)
751 VB=(VFACT/DEN)*10.0**M
752 MB=Z(NPOINT)/DFLOAT(INC)
753 C *****
754 C
755 THE FOLLOWING SECTION SETS UP THE PIECEWISE LINEAR AND RUNGE-KUTTA
756 RESULTS SO THAT 26 POINTS OF THE PIECEWISE SOLUTION AND 27 OF THE
757 RUNGE-KUTTA VALUES ARE PLOTTED FOR A CLEARER REPRESENTATION, WITH
758 THE PLOTS BEING ALTERNATED FOR COMPARISON PURPOSES.
759 C
760 YP=ARRAY FOR PLOTTED PIECEWISE LINEAR DISPLACEMENTS
761 ZP=ARRAY FOR PLOTTED PIECEWISE LINEAR Z VALUES
762 YRK=ARRAY FOR PLOTTED RUNGE-KUTTA DISPLACEMENTS
763 ZPRK=ARRAY FOR PLOTTED RUNGE-KUTTA Z VALUES
764 C
765 C
766 C**NOTE**=THE NUMBER OF POINTS TO BE PLOTTED MUST NOT EXCEED 1000.
767 C
768 C *****
769 INCR=NPOINT/25
770 INCRRK=INCR/2

```



```

771      NPP=1
772      NPRK=1
773      NDP=1
774      NDRK=1
775 740 CONTINUE
776      YP(NPP)=AMPLP(NDP)
777      ZP(NPP)=Z(NDP)
778      IF(NDP.EQ.NPOINT) GO TO 750
779      NPP=NPP+1
780      NDP=NDP+INCR
781      GO TO 740
782 750 CONTINUE
783      YRK(NPRK)=AMPLRK(NDRK)
784      ZPRK(NPRK)=Z(NDRK)
785      IF(NDRK.EQ.1) GO TO 760
786      NDRK=NDRK+INCR
787      GO TO 770
788 760 CONTINUE
789      NDRK=NDRK+INCRRK
790 770 CONTINUE
791      IF(NDRK.GT.NPOINT) GO TO 780
792      NPRK=NPRK+1
793      GO TO 750
794 780 CONTINUE
795      NPRK=NPRK+1
796      YRK(NPRK)=AMPLRK(NPOINT)
797      ZPRK(NPRK)=Z(NPOINT)
798 C*****
799 C
800 C      PLOTTING THE SOLUTIONS.  CONSULT THE WRITEUP ON CGPL/CGPL2 AND
801 C      THE MANUAL ON DIGITAL PLOTTING FOR DETAILS.
802 C
803 C*****
804      NF=1
805      ND=NPOINT
806      CALL CGPL2(Z,AMPLS,ND,NF,5,HA,HB,HC,VA,VB,VC,ALPH)
807      ND=NPP
808      CALL CGPL2(ZP,YP,ND,2,1,HA,HB,HC,VA,VB,VC,ALPH)
809      ND=NPRK
810      CALL CGPL2(ZPRK,YRK,ND,3,1,HA,HB,HC,VA,VB,VC,ALPH)
811      NF=4
812      CALL CGPL2(XA,YA,2,NF,4,HA,HB,HC,VA,VB,VC,ALPH)
813      HORIZ=4.15
814      VERT=-0.475
815      CALL SYMBOL(HORIZ,VERT,0.2,163,0.0,-1)
816      HORIZ=HA
817      VERT=VC+0.5
818      CALL PLOT(HORIZ,VERT,3)
819      HORIZ=HC
820      CALL PLOT(HORIZ,VERT,2)
821      VERT=VERT+4.0
822      CALL PLOT(HORIZ,VERT,2)
823      HORIZ=HA
824      CALL PLOT(HORIZ,VERT,2)
825      VERT=VC+0.5
826      CALL PLOT(HORIZ,VERT,2)
827      STARTX=(HC-7.7)/2.0
828      STARTY=VC+0.5
829      HORIZ=STARTX+0.2
830      VERT=STARTY+3.6
831      CALL SYMBOL(HORIZ,VERT,0.2,'CHAR. NUM. (EST. VALUE) =',0.0,25)
832      HORIZ=STARTX+5.4
833      CALL NUMBER(HORIZ,VERT,0.2,AGUESS,0.0,4)
834      HORIZ=STARTX+0.2
835      VERT=STARTY+3.2
836      CALL SYMBOL(HORIZ,VERT,0.2,'CHAR. NUM. (FINAL VALUE) =',0.0,26)
837      HORIZ=STARTX+5.6
838      CALL NUMBER(HORIZ,VERT,0.2,A,0.0,4)
839      HORIZ=STARTX+0.2
840      VERT=STARTY+2.8
841      CALL SYMBOL(HORIZ,VERT,0.2,'Q =',0.0,3)
842      HORIZ=STARTX+1.0
843      CALL NUMBER(HORIZ,VERT,0.2,Q,0.0,4)
844      HORIZ=STARTX+3.5
845      CALL SYMBOL(HORIZ,VERT,0.2,S2,0.0,-1)
846      HORIZ=STARTX+3.7
847      CALL SYMBOL(HORIZ,VERT,0.2,'Z =',0.0,3)
848      HORIZ=STARTX+4.5
849      CALL SYMBOL(HORIZ,VERT,0.2,163,0.0,-1)
850      HORIZ=STARTX+4.7
851      CALL SYMBOL(HORIZ,VERT,0.2,97,0.0,-1)
852      HORIZ=STARTX+4.9
853      CALL NUMBER(HORIZ,VERT,0.2,H,0.0,1)
854      HORIZ=STARTX+0.2
855      VERT=STARTY+2.0
856      CALL SYMBOL(HORIZ,VERT,0.2,'Y(O) =',0.0,6)
857      HORIZ=STARTX+1.6
858      CALL NUMBER(HORIZ,VERT,0.2,DISPLO,0.0,4)
859      HORIZ=STARTX+3.5
860      CALL SYMBOL(HORIZ,VERT,0.2,'Y',0.0,1)
861      HORIZ=STARTX+3.7
862      CALL SYMBOL(HORIZ,VERT,0.2,125,0.0,-1)
863      HORIZ=STARTX+3.9
864      CALL SYMBOL(HORIZ,VERT,0.2,'(O) =',0.0,5)
865      HORIZ=STARTX+5.1
866      CALL NUMBER(HORIZ,VERT,0.2,VO,0.0,4)
867      HORIZ=STARTX+0.3
868      VERT=STARTY+1.1
869      CALL PLOT(HORIZ,VERT,3)
870      HORIZ=STARTX+0.8
871      CALL PLOT(HORIZ,VERT,2)
872      HORIZ=STARTX+1.0
873      VERT=STARTY+1.0
874      CALL SYMBOL(HORIZ,VERT,0.2,'SERIES APPROXIMATION',0.0,20)
875      HORIZ=STARTX+0.6
876      VERT=STARTY+0.7
877      CALL SYMBOL(HORIZ,VERT,0.2,1,0.0,-1)
878      HORIZ=STARTX+1.0
879      VERT=STARTY+0.6
880      CALL SYMBOL(HORIZ,VERT,0.2,'PIECEWISE LINEARIZATION',0.0,23)

```



```

881      HORIZ=STARTX+0.6
882      VERT=STARTY+0.3
883      CALL SYMBOL(HORIZ,VERT,0.2,2,0.0,-1)
884      HORIZ=STARTX+1.0
885      VERT=STARTY+0.2
886      CALL SYMBOL(HORIZ,VERT,0.2,'RUNGE-KUTTA',0.0,11)
887      NF=0
888      CALL CGPL2(XA,YA,2,NF,4,HA,HB,HC,VA,VB,VC,ALPH)
889      STOP
890      END
891
892 C
893 C
894 C
895 C
896 C
897 C
898 C
899 C
900 C
901 C
902 C
903 C      THE FOLLOWING SUBROUTINE CALCULATES THE OUTPUT VALUE FOR A GUESS
904 C      FOR THE CHARACTERISTIC NUMBER, USING THE RECURRENCE RELATIONS THAT
905 C      ARE DESCRIBED IN MCLACHLAN AND INCE.
906 C
907 C
908 C      SUBROUTINE RECUR
909 C      IMPLICIT REAL=8 (A-H,O-Z)
910 C      COMMON/CALC/AIN,DISPLO,DIFF,AOUT,IPAR
911 C      COMMON/PARAM/Q
912 C
913 C
914 C      R=COUNTER FOR THE NUMBER OF ITERATIONS FOR THE PARTICULAR RELATION
915 C      USED
916 C      GAMMAR=FACTOR USED IN THE CONTINUED FRACTION (SEE INCE)
917 C
918 C      FOR NON-ZERO INITIAL DISPLACEMENT AND EVEN ORDER, THE PROGRAM
919 C      MOVES TO STEP 800. FOR ZERO INITIAL DISPLACEMENT AND EVEN ORDER,
920 C      IT GOES TO STEP 820. IF THE ORDER IS ODD AND INITIAL DISPLACEMENT
921 C      NON-ZERO, IT CONTINUES AT STEP 840, AND IF THE DISPLACEMENT
922 C      HAPPENS TO BE ZERO, IT GOES TO STEP 860.
923 C
924 C
925 C      R=200.0
926 C      GAMMAR=0.0
927 C      IF((DISPLO.NE.0.0).AND.(IPAR.EQ.1)) GO TO 800
928 C      IF((DISPLO.EQ.0.0).AND.(IPAR.EQ.1)) GO TO 820
929 C      IF((DISPLO.NE.0.0).AND.(IPAR.EQ.0)) GO TO 840
930 C      GO TO 860
931 C
932 C
933 C      THE CASE OF NON-ZERO INITIAL DISPLACEMENT AND EVEN ORDER.
934 C
935 C      PHIR,DENOMR=FACTORS USED IN CALCULATING CONTINUED FRACTION
936 C      A=VALUE OBTAINED
937 C
938 C
939 C      800 CONTINUE
940 C      PHIR=1.0-AIN/(4.0-R**2)-GAMMAR
941 C      DENOMR=(Q**2)/((16.0-R**2)*(R-1.0)**2)
942 C      GAMMAR=DENOMR/PHIR
943 C      R=R-1.0
944 C      IF(R.EQ.1.0) GO TO 810
945 C      GO TO 800
946 C
947 C      810 CONTINUE
948 C      A=-(Q**2)/(2.0*(1.0-AIN/4.0-GAMMAR))
949 C      GO TO 870
950 C
951 C
952 C      ZERO INITIAL DISPLACEMENT AND EVEN ORDER.
953 C
954 C
955 C      820 CONTINUE
956 C      PHIR=1.0-AIN/(4.0-R**2)-GAMMAR
957 C      DENOMR=(Q**2)/((16.0-R**2)*(R-1.0)**2)
958 C      GAMMAR=DENOMR/PHIR
959 C      R=R-1.0
960 C      IF(R.EQ.2.0) GO TO 830
961 C      GO TO 820
962 C
963 C      830 CONTINUE
964 C      A=4.0-(Q**2)/(16.0*(1.0-AIN/16.0-GAMMAR))
965 C      GO TO 870
966 C
967 C
968 C      THE SITUATION OF ODD ORDER AND NON-ZERO INITIAL DISPLACEMENT
969 C
970 C
971 C      840 CONTINUE
972 C      PHIR=1.0-AIN/(2.0-R+1.0)**2-GAMMAR
973 C      DENOMR=(Q**2)/((4.0-R**2-1.0)**2)
974 C      GAMMAR=DENOMR/PHIR
975 C      IF(R.EQ.1.0) GO TO 850
976 C      R=R-1.0
977 C      GO TO 840
978 C
979 C      850 CONTINUE
980 C      A=1.0+Q-GAMMAR
981 C      IF(Q.LT.0.0) Q=DABS(Q)
982 C      GO TO 870
983 C
984 C
985 C      THE CASE OF ODD ORDER AND ZERO INITIAL DISPLACEMENT.
986 C
987 C
988 C      860 CONTINUE
989 C      Q=-Q
990 C      GO TO 840
991 C
992 C      870 CONTINUE
993 C      AOUT=A
994 C
995 C

```



```

991      C
992      C      DIFF=DIFFERENCE BETWEEN INPUT AND OUTPUT VALUES (USED IN
993      C      TERMINATING ITERATIONS FOR CHARACTERISTIC NUMBER)
994      C
995      C *****
996      C      DIFF=AIN-AOUT
997      C      RETURN
998      C      END
999      E
1000     E
1001     E
1002     E *****
1003     E
1004     E      THE FOLLOWING SUBROUTINE CONTAINS THE DIFFERENTIAL EQUATIONS TO BE
1005     E      SOLVED BY DYERK: THE INSTANTANEOUS VELOCITY AND THE ORIGINAL
1006     E      EQUATION.
1007     E
1008     C *****
1009     C      SUBROUTINE FCN(NEON,ZRK,Y,YPRIME)
1010     C      IMPLICIT REAL*8 (A-H,O-Z)
1011     C      REAL*8 Y(NEON),YPRIME(NEON),ZRK
1012     C      COMMON/CHAR/A
1013     C      COMMON/PARAM/Q
1014     C      YPRIME(1)=Y(2)
1015     C      YPRIME(2)=- (A-2.*Q*DCOS(2.*ZRK))*Y(1)
1016     C      RETURN
1017     C      END

```

END OF FILE

B30348